In Taiwan's mountains, a laboratory for landslides p. 938

Tomorrow's Earth pp. 950 & 969

Enhanced resolution of chromatin wins *Science* & SciLifeLab Prize p. 964

Science

$15
22 NOVEMBER 2019
sciencemag.org

AAAS

BEETLE HORNS

Beetle thoracic horns share a developmental origin with wings pp. 946 & 1004

COMPUTER SCIENCE

Preventing undesirable behavior of intelligent machines

Philip S. Thomas[1]*, Bruno Castro da Silva[2], Andrew G. Barto[1], Stephen Giguere[1], Yuriy Brun[1], Emma Brunskill[3]

Intelligent machines using machine learning algorithms are ubiquitous, ranging from simple data analysis and pattern recognition tools to complex systems that achieve superhuman performance on various tasks. Ensuring that they do not exhibit undesirable behavior—that they do not, for example, cause harm to humans—is therefore a pressing problem. We propose a general and flexible framework for designing machine learning algorithms. This framework simplifies the problem of specifying and regulating undesirable behavior. To show the viability of this framework, we used it to create machine learning algorithms that precluded the dangerous behavior caused by standard machine learning algorithms in our experiments. Our framework for designing machine learning algorithms simplifies the safe and responsible application of machine learning.

Machine learning (ML) algorithms are having an increasing impact on modern society. They are used by geologists to predict landslides (*1*) and by biologists working to create a vaccine for HIV (*2*); they also influence criminal sentencing (*3*), control autonomous vehicles (*4*), and enable medical advances (*5*). The potential for ML algorithms to cause harm—including catastrophic harm—is therefore a pressing concern (*6*). Despite the importance of this problem, current ML algorithms do not provide their users with an effective means for precluding undesirable behavior, which makes the safe and responsible use of ML algorithms difficult. We introduce a framework for designing ML algorithms that allow their users to easily define and regulate undesirable behavior. This framework does not address the problem of imbuing intelligent machines with a notion of morality or human-like values (*7*), nor the problem of avoiding undesirable behavior that the user never considered (*8*). Rather, it provides a remedy for the problem of ML algorithms that exhibit undesirable behavior because their users did not have an effective way to specify and constrain such behavior.

The first step of the current standard approach for designing ML algorithms, which we refer to as the standard ML approach, is to define mathematically what the algorithm should do. At an abstract level, this definition is the same across all branches of ML: Find a solution θ^*, within a feasible set Θ, that maximizes an objective function $f\colon \Theta \to \mathbb{R}$. That is, the goal of the algorithm is to find a solution in

$$\arg\max_{\theta \in \Theta} f(\theta) \tag{1}$$

Note that the algorithm does not know $f(\theta)$ for any $\theta \in \Theta$ (e.g., the true mean squared error); it can only reason about it from data (e.g., by using the sample mean squared error).

One problem with the standard ML approach is that the user of an ML algorithm must encode constraints on the algorithm's behavior in the feasible set or the objective function. Encoding constraints in the objective function [e.g., using soft constraints (*9*) or robust and risk-sensitive approaches (*10*)] requires extensive domain knowledge or additional data analysis to properly balance the relative importance of the primary objective function and the constraints. Similarly, encoding constraints in the feasible set [e.g., using hard constraints (*9*), chance constraints (*11*), or robust optimization approaches (*12*)] requires knowledge of the probability distribution from which the available data are sampled, which is often not available.

Our framework for designing ML algorithms allows the user to constrain the behavior of the algorithm more easily, without requiring extensive domain knowledge or additional data analysis. This is achieved by shifting the burden of ensuring that the algorithm is well-behaved from the user of the algorithm to the designer of the algorithm. This is important because ML algorithms are used for critical applications by people who are experts in their fields, but who may not be experts in ML and statistics.

We now define our framework. Let D, called the data, be the input to the ML algorithm. For example, in the classification setting, D is not a single labeled training example but rather all of the available labeled training examples. D is a random variable and the source of randomness in our subsequent statements regarding probability. An ML algorithm is a function a, where $a(D)$ is the solution output by the algorithm when trained on data D. Let Θ be the set of all possible solutions that an ML algorithm could output. Our framework mathematically defines what an algorithm should do in a way that allows the user to directly place probabilistic constraints on the solution, $a(D)$, returned by the algorithm. This differs from the standard ML approach wherein the user can only indirectly constrain $a(D)$ by restricting or modifying the feasible set Θ or objective function f. Concretely, algorithms constructed using our framework are designed to satisfy constraints of the form $\Pr(g(a(D)) \leq 0) \geq 1 - \delta$, where $g\colon \Theta \to \mathbb{R}$ defines a measure of undesirable behavior (as illustrated later by example) and $\delta \in [0, 1]$ limits the admissible probability of undesirable behavior.

Note that in these constraints, D is the only source of randomness; we denote random variables by capital noncalligraphic letters to make clear which terms are random in statements of probability and expectation. Because these constraints define which algorithms a are acceptable (rather than which solutions θ are acceptable), they must be satisfied during the design of the algorithm rather than when the algorithm is applied. This shifts the burden of ensuring that the algorithm is well-behaved from the user to the designer.

Using our framework for designing ML algorithms involves three steps:

1) Define the goal for the algorithm design process. The designer of the algorithm writes a mathematical expression that expresses a goal—in particular, the properties that the designer wants the resulting algorithm a to have. This expression has the following form, which we call a Seldonian optimization problem after a fictional character (*13*):

$$\arg\max_{a \in \mathcal{A}} f(a)$$
$$\text{s.t. } \forall i \in \{1, \ldots, n\}, \Pr\Big(g_i(a(D)) \leq 0\Big) \geq 1 - \delta_i \tag{2}$$

where $\mathcal{A}$ is the set of all algorithms that will be considered by the designer, $f\colon \mathcal{A} \to \mathbb{R}$ is now an objective function that quantifies the utility of an algorithm, and we allow for $n \geq 0$ constraints, each defined by a tuple (g_i, δ_i), where $i \in \{1, \ldots, n\}$. Note that this is in contrast to the standard ML approach: In the standard ML approach, Eq. 1 defines the goal of the algorithm, which is to produce a solution with a given set of properties, whereas in our framework, Eq. 2 defines the goal of the designer, which is to produce an algorithm with a given set of properties.

2) Define the interface that the user will use. The user should have the freedom to specify one or more g_i that capture the user's own definition of undesirable behavior. This requires the algorithm a to be compatible with many different definitions of g_i. The designer should therefore specify the class of possible definitions of g_i with which the algorithm will be

[1]University of Massachusetts, Amherst, MA, USA. [2]Universidade Federal do Rio Grande do Sul, Porto Alegre, Rio Grande do Sul, Brazil. [3]Stanford University, Stanford, CA, USA. *Corresponding author. Email: pthomas@cs.umass.edu

compatible, and should provide a means for the user to tell the algorithm which definition of g_i should be used, without requiring the user to have knowledge of the distribution of D or even the value $g_i(\theta)$ for any $\theta \in \Theta$. Below, we provide examples of how this can be achieved.

3) Create the algorithm. The designer creates an algorithm a, which is a (possibly approximate) solution to Eq. 2 from step 1 and which allows for the class of g_i chosen in step 2. In practice, designers rarely produce algorithms that cannot be improved upon, which implies that they may only find approximate solutions to Eq. 2. Our framework allows for this by requiring a to satisfy only the probabilistic constraints while attempting to optimize f; we call such algorithms Seldonian. We call an algorithm quasi-Seldonian if it relies on reasonable but false assumptions, such as appeals to the central limit theorem. See (*14*) for further discussion regarding the benefits and limitations of quasi-Seldonian algorithms.

Once a Seldonian algorithm has been designed, a user can apply it by specifying one or more g_i (belonging to the class of g_i chosen in step 2 above) to capture the user's desired definition of undesirable behavior, and specifying δ_i, the maximum admissible probability of the undesirable behavior characterized by g_i.

To show the viability of our framework, we used it to design regression, classification, and reinforcement learning algorithms. Constraining the behavior of regression and classification algorithms is important because, for example, they have been used for medical applications where undesirable behavior could delay cancer diagnoses (*15*), and because they have been shown to sometimes cause racist, sexist, and other discriminatory behavior (*3*, *16*). Similarly, reinforcement learning algorithms have been proposed for applications where undesirable behavior can cause financial losses (*17*), environmental damage (*18*), and even death (*19*). The Seldonian algorithms and applications we present below are illustrations to show that it is possible and tractable to design Seldonian algorithms that can tackle important problems of interest. Note that these are intended only as proof of principle; the primary contribution of this work is the framework itself rather than any specific algorithm or application. Like the common application of classification algorithms (*20*) to the Wisconsin breast cancer dataset (*21*), the applications validate our ML algorithms as tools that researchers with medical expertise and domain knowledge could apply (*22*), but do not imply that our learned solutions (classifiers or policies) should be deployed as-is to any particular real-world problem.

The regression algorithm that we designed attempts to minimize the mean squared error of its predictions while ensuring that, with high probability, a statistic of interest, $g(\theta)$, of the returned solution, $\theta = a(D)$, is bounded. The definition of this statistic can be chosen by the user to capture a particular definition of undesirable behavior (e.g., the expected financial loss that results from using a given solution θ). The user may not know the value of this statistic for even a single solution. We must therefore provide the user with a way to tell our algorithm the statistic to be bounded, without requiring the user to provide the value, $g(\theta)$, of the statistic for different solutions θ (see step 2 above). To achieve this (*14*), we allow the user to specify a sample statistic $\hat{g}(\theta, D)$, and we define $g(\theta)$ to be the expected value of this sample statistic: $g(\theta) = \mathbf{E}[\hat{g}(\theta, D)]$, where $\mathbf{E}$ denotes expected value.

The creation of a regression algorithm (step 3) with the properties specified during steps 1 and 2 is challenging. This is to be expected given the shifted burden discussed previously; see (*14*) for a detailed description of how we performed step 3 when designing all of the Seldonian algorithms that we present. Figure 1 overviews our regression algorithms.

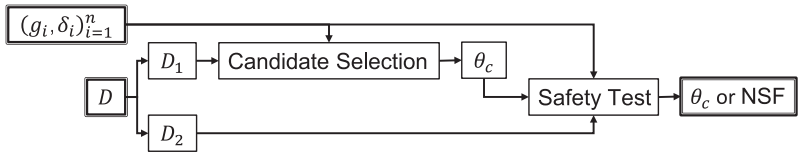


Fig. 1. Overview of Seldonian regression algorithms. The algorithm takes the behavioral constraints $(g_i, \delta_i)_{i=1}^n$ and training data D as input and outputs either a solution θ_c or NSF (no solution found). First, the data are partitioned into two sets, D_1 and D_2. Next, a routine called Candidate Selection uses D_1 to select a single solution, the candidate solution θ_c, which it predicts will perform well under the primary objective f while also being likely to pass the subsequent safety test based on knowledge of the specific form of the test. The Safety Test mechanism checks whether the algorithm has sufficient confidence that $g_i(\theta_c) \leq 0$ for each constraint $i \in \{1, ..., n\}$. If so, it returns the candidate solution θ_c, otherwise it returns NSF. The Safety Test routine uses standard statistical tools such as Student's t test and Hoeffding's inequality to transform sample statistics computed from D_2 into bounds on the probability that $g(a(D)) > 0$ (i.e., bounds on the probability of undesirable behavior).

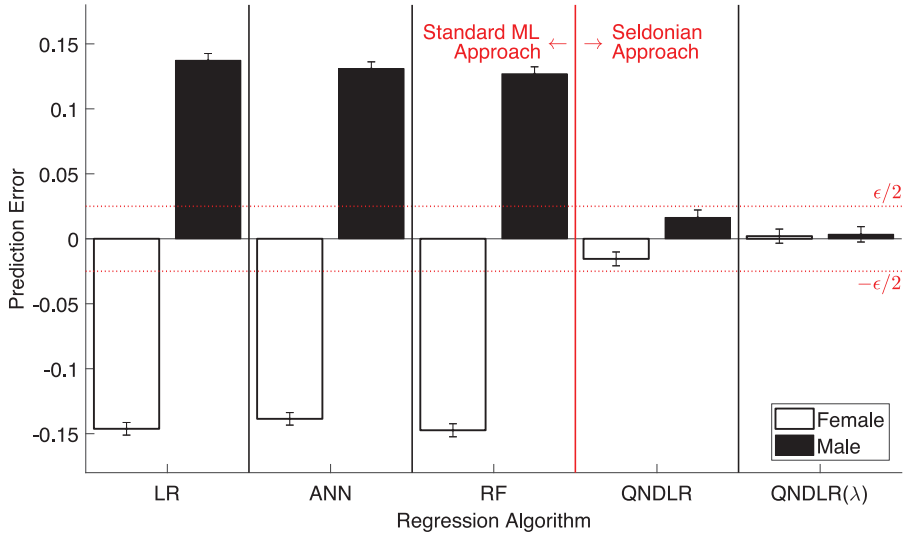


Fig. 2. Seldonian regression algorithm applied to GPA prediction. We used five different regression algorithms to predict students' GPAs during their first three semesters at university based on their scores on nine entrance exams. We used actual data from 43,303 students from Brazil. Here, the user-selected definition of undesirable behavior corresponds to large differences in mean prediction errors (mean predicted GPA minus mean observed GPA) for applicants of different genders. This plot shows the mean prediction errors (±SD) for male and female students when using each regression algorithm. We used three standard ML algorithms—least squares linear regression (LR) (*40*), an artificial neural network (ANN) (*41*), and a random forest (RF) (*42*)—and two variants of our Seldonian algorithm: QNDLR and QNDLR(λ). All shown standard ML methods tend to notably overpredict the performance of male students and underpredict the performance of female students, whereas the two variants of our Seldonian regression algorithm do not. In particular, our algorithms ensure that, with approximately 95% probability, the expected prediction errors for men and women will be within $\varepsilon = 0.05$, and both effectively preclude the sexist behavior that was exhibited by the standard ML algorithms.

Recent methods designed particularly for algorithmic fairness in regression tasks (*23*), developed in parallel to our own, do not give users the freedom to select their own desired definitions of undesirable behavior, nor do they provide guarantees on the avoidance of such behavior.

We applied a variant of our Seldonian regression algorithm to the problem of predicting students' grade point averages (GPAs) during their first three semesters at university on the basis of their scores on nine entrance exams; we used a sample statistic that captures one form of discrimination (sexism). Note that our algorithm is not particular to the chosen measure of discrimination; see (*14*) for a discussion of other definitions of fairness. Figure 2 presents the results of this experiment, showing that commonly used regression algorithms designed using the standard ML approach can discriminate against female

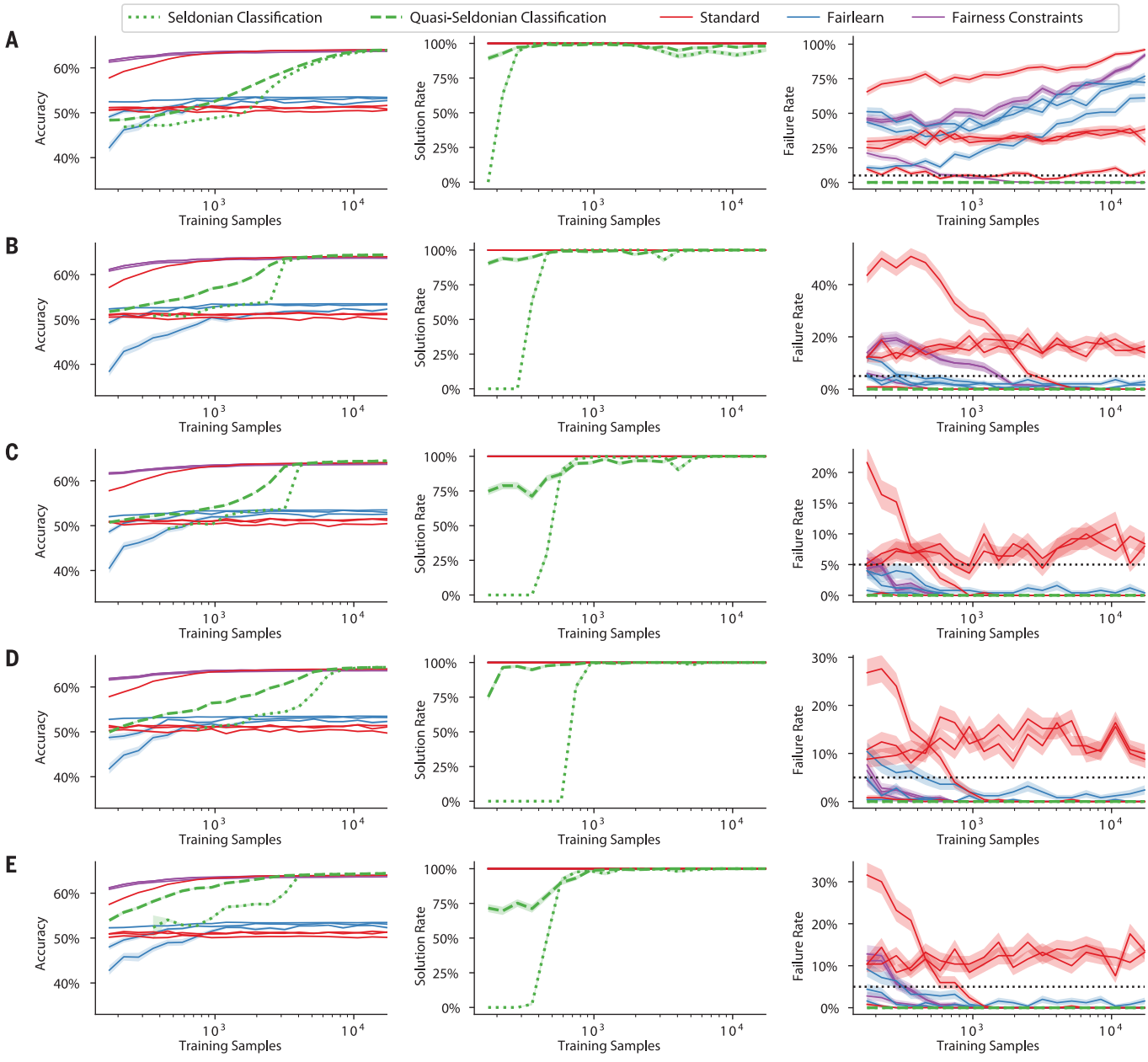


Fig. 3. Seldonian classification algorithm applied to GPA prediction. We applied classification algorithms to predict whether student GPAs will be above 3.0. Shaded regions represent SE over 250 trials. The curves labeled "Standard" correspond to common classification algorithms designed using the standard ML approach; the multiple curves for Fairlearn and Fairness Constraints correspond to different hyperparameter settings for each algorithm (*14*). Each row corresponds to a different fairness definition: (**A**) disparate impact, (**B**) demographic parity, (**C**) equal opportunity, (**D**) equalized odds, (**E**) predictive equality. The horizontal axes of all plots correspond to the amount of training data and have logarithmic scale. The left column shows the accuracy of the trained classifiers, the center column shows the probability that each algorithm returned a solution (non-Seldonian algorithms always returned solutions), and the right column shows the probability that each classifier violated a behavioral constraint. When showing the failure rate of each algorithm, the horizontal dashed line corresponds to 100δ%, where δ = 0.05. In all cases, the Seldonian and quasi-Seldonian algorithms returned solutions using a reasonable amount of data (center), did so without significant losses to accuracy (left), and were the only algorithms to reliably enforce all five fairness definitions (right).

students when applied without considerations for fairness. In contrast, the user can easily limit the observed sexist behavior in Fig. 2 using our Seldonian regression algorithm.

To emphasize that Seldonian algorithms are compatible with a variety of definitions of fairness and to better situate our research relative to current state-of-the-art fairness-aware ML algorithms, we present a Seldonian classification algorithm (*14*). This classification algorithm differs from our regression algorithm in its primary objective (classification loss rather than mean squared error) and in its more sophisticated interface, which allows the user to type an expression that defines $g(\theta)$ in terms of common statistics (such as the false negative rate or false positive rate given that the protected attribute, here gender, takes a specific value), constants, operators (such as addition, division, and absolute value), and statistics for which the user can provide unbiased estimates, as in the regression example. We applied our classification algorithm to predicting whether student GPAs will be above 3.0 using the dataset described in Fig. 2, while constraining five popular definitions of fairness for classification (Fig. 3). The Seldonian classification algorithm properly limited the specified form of unfair behavior across all trials. Unlike our approach, fairness-aware classification algorithms designed using the standard ML approach do not provide probabilistic guarantees that the resulting classifier is acceptably fair when applied to unseen data. We observed that two state-of-the-art fairness-aware algorithms that we ran for comparison, Fairlearn (*24*) and Fairness Constraints (*25*), each produced unfair behavior under at least one definition of fairness.

Next, we used our framework to design a general-purpose Seldonian reinforcement learning algorithm: one that, unlike regression and classification algorithms, makes a sequence of dependent decisions. In this context, a solution θ is called a policy; a history H (a random variable) denotes the outcome of using a policy to make a sequence of decisions; and the available data D is a set of histories produced by some initial policy θ_0. Because it is Seldonian, our algorithm searches for an optimal policy while ensuring that $\Pr(g(a(D)) \leq 0) \geq 1 - \delta$. The algorithm we designed is compatible with g of the form $g(\theta) = \mathbf{E}[r'(H)|\theta_0] - \mathbf{E}[r'(H)|\theta]$, where the user selects $-r'(H)$ to measure a

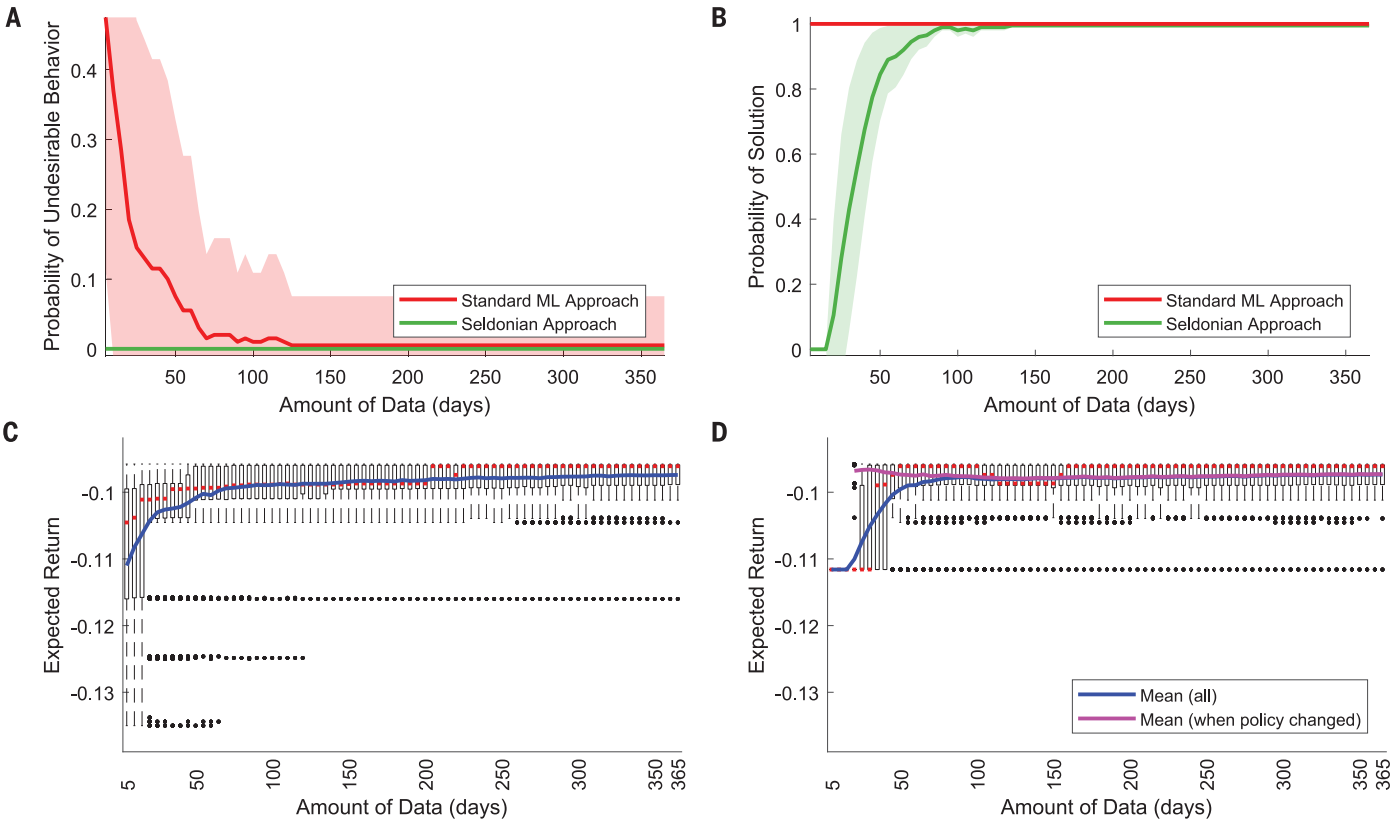


Fig. 4. Seldonian reinforcement learning algorithm for proof-of-principle bolus calculation in type 1 diabetes. Results are averaged over 200 trials; shaded regions denote SE. The Seldonian algorithm is compared to an algorithm built using the standard ML approach that penalizes the prevalence of low blood sugar. (**A**) Probability that each method returns policies (solutions) that increase the prevalence of low blood sugar. The algorithm designed using the standard ML approach often proposed policies that increased the prevalence of low blood sugar, violating the safety constraint, even though it used an objective function (reward function) that penalized instances of hypoglycemia. In contrast, across all trials, our Seldonian algorithm was safe; it never changed the treatment policy in a way that increased the prevalence of low blood sugar. (**B**) Probability that each method returns a policy that differs from the initial policy. Our Seldonian algorithm was able to safely improve upon the initial policy with just 1 to 5 months of data. (**C**) Box plot (with outliers plotted) of the distribution of the expected returns (objective function values) of the treatment policies returned by the standard ML algorithm. The blue line depicts the sample mean; red lines within the boxes mark the medians. All points below −0.1116 [where the blue curve in (D) begins] correspond to cases where the standard ML algorithm both decreased performance and produced undesirable behavior (an increase in the prevalence of low blood sugar). (**D**) Similar to (C), but showing results for the Seldonian algorithm. The magenta line is the average of the performance when the algorithm produced a policy that differed from the initial policy. Notice that all points have values of at least −0.1116, indicating that our algorithm never produced undesirable behavior. When boxes appear to be missing, the boxes have zero width and are obscured by the red line indicating the median of the box.

particular definition of how undesirable the history H is. That is, with probability at least $1 - \delta$, the algorithm will not output a policy θ that increases the user-specified measure of undesirable behavior. Notice that the user need only be able to recognize undesirable behavior to define r'; the user does not need to know the distributions over histories H that result from applying different policies. For example, the user might define $r'(H) = -1$ if undesirable behavior occurred in H, and $r'(H) = 0$ otherwise.

Some previous reinforcement learning methods are guaranteed to increase the primary objective with high probability (*26*–*28*). These algorithms can be viewed as Seldonian or quasi-Seldonian algorithms that are restricted to only work with one definition of undesirable behavior: a decrease in the primary objective. This restricted definition of undesirable behavior precludes their application to problems where undesirable behavior does not align perfectly with the primary objective (see fig. S31 for an example where the behavioral constraint and primary objective are conflicting). Similarly, data-driven robust optimization (*29*) has also provided high-probability guarantees on constraint satisfaction, but only for convex constraints and a subset of objectives f that do not include the regression, classification, and reinforcement learning examples we consider (*14*).

Of the many high-risk, high-reward applications of reinforcement learning that have been proposed, we selected one to show the feasibility of our approach: automatically adjusting the treatment for a person with type 1 diabetes (*30*, *31*). In this application, a policy θ (as defined above) is a bolus calculator, which determines the amount of insulin that a person should inject prior to ingestion of a carbohydrate-containing meal to avoid high blood sugar levels. To simulate the metabolism of a human, we used a detailed metabolic simulator (*32*). Each history H corresponds to the outcome of 1 day, and we defined $-r'(H)$ to be a measure of the prevalence of low blood sugar (with particularly large penalties for hypoglycemia, i.e., dangerously low blood sugar levels) in the history H. Enforcing high-probability safety constraints on hypoglycemia is important because of the severe health consequences caused by hypoglycemia, including altered mental status, confusion, coma, and even death (*33*–*35*).

Figure 4 shows the result of applying both our Seldonian algorithm and a baseline algorithm designed using the standard ML approach. The baseline algorithm uses a technique called importance sampling (*36*) to estimate the performance of all policies using the data D generated by the initial policy θ_0, and it returns the policy predicted to perform best. This non-Seldonian algorithm (*14*) closely resembles our Seldonian algorithm with the behavioral constraints removed. Neither the Seldonian algorithm nor the corresponding standard ML approach algorithm are meant to be used directly in clinical practice; however, comparing their behavior provides insight into the effects of our Seldonian framework. Note from Fig. 4 that our algorithm does not propose a new policy until it has high confidence that the prevalence of low blood sugar will not increase. Our algorithm is not specific to this particular choice of constraint [see (*14*) for implementation of alternative constraints, such as constraints on the mean time hyperglycemic]. Our approach is complementary to existing work on personalized bolus calculators that do not use reinforcement learning but rely on experts or prior data to set critical parameters (*14*, *37*). These parameters could be adapted for each individual using a reinforcement learning approach, and a Seldonian reinforcement learning algorithm would ensure that it would alter the parameters only when it is highly confident that the change would not cause undesirable behavior (e.g., increase the prevalence of hypoglycemia) for the particular individual. Although any clinical application would leverage a more complicated policy than what we consider here, we use this as an illustration of how a Seldonian algorithm could be used as part of a broader effort to provide personalized policies for high-stakes applications.

Given the recent rise of real-world ML applications and the corresponding surge of potential harm that they could cause, it is imperative that ML algorithms provide their users with an effective means for controlling behavior. To this end, we have proposed a framework for designing ML algorithms and shown how it can be used to construct algorithms that provide their users with the ability to easily (that is, without requiring additional data analysis) place limits on the probability that the algorithm will produce any specified undesirable behavior. Algorithms designed using our framework are not just a replacement for ML algorithms in existing applications; it is our hope that they will pave the way for new applications for which the use of ML was previously deemed to be too risky.

REFERENCES AND NOTES

1. R. W. Jibson, *Eng. Geol.* **91**, 209–218 (2007).
2. M. Bhasin, G. Raghava, *Vaccine* **22**, 3195–3204 (2004).
3. J. Angwin, J. Larson, S. Mattu, L. Kirchner, Machine bias. *ProPublica*, May 2016; www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.
4. D. A. Pomerleau, *Adv. Neural Inform. Process. Syst.* **1**, 305–313 (1988).
5. S. Saria, *IEEE Intell. Syst.* **29**, 82–87 (2014).
6. N. Bostrom, *Superintelligence: Paths, Dangers, Strategies* (Oxford Univ. Press, 2014).
7. S. Russell, *Sci. Am.* **314**, 58–59 (June 2016).
8. D. Amodei *et al.*, arXiv 1606.06565 [cs.AI] (25 July 2016).
9. S. Boyd, L. Vandenberghe, *Convex Optimization* (Cambridge Univ. Press, 2004).
10. D. Bertsimas, G. J. Lauprete, A. Samarov, *J. Econ. Dyn. Control* **28**, 1353–1381 (2004).
11. A. Charnes, W. W. Cooper, *Manage. Sci.* **6**, 73–79 (1959).
12. A. Ben-Tal, L. El Ghaoui, A. Nemirovski, *Robust Optimization* (Princeton Univ. Press, 2009).
13. I. Asimov, *Foundation* (Gnome, 1951).
14. See supplementary materials.
15. O. L. Mangasarian, W. N. Street, W. H. Wolberg, *Oper. Res.* **43**, 570–577 (1995).
16. L. Weber, "Your résumé vs. oblivion." *Wall Street Journal* (2012); www.wsj.com/articles/SB10001424052970204624204577178941034941330.
17. L. Li, W. Chu, J. Langford, R. E. Schapire, A contextual-bandit approach to personalized news article recommendation. In *International World Wide Web Conference* (2010), pp. 661–670.
18. R. M. Houtman *et al.*, *Int. J. Wildland Fire* **22**, 871–882 (2013).
19. B. Moore, P. Panousis, V. Kulkarni, L. Pyeatt, A. Doufas, Reinforcement learning for closed-loop propofol anesthesia: A human volunteer study. In *Proceedings of the Twenty-Second Innovative Applications of Artificial Intelligence Conference* (2010), pp. 1807–1813; www.aaai.org/ocs/index.php/IAAI/IAAI10/paper/view/1572/2359.
20. K. Grabczewski, W. Duch, Heterogeneous forests of decision trees. In *International Conference on Artificial Neural Networks* (2002), pp. 504–509.
21. D. Dheeru, E. Karra Taniskidou, UCI Machine Learning Repository (2017); http://archive.ics.uci.edu/ml.
22. K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, D. I. Fotiadis, *Comput. Struct. Biotechnol. J.* **13**, 8–17 (2015).
23. J. Komiyama, A. Takeda, J. Honda, H. Shimao, *Proc. Mach. Learn. Res.* **80**, 2737–2746 (2018).
24. A. Agarwal, A. Beygelzimer, M. Dudík, J. Langford, H. Wallach, *Proc. Mach. Learn. Res.* **80**, 60–69 (2018).
25. M. B. Zafar, I. Valera, M. G. Rodriguez, K. P. Gummadi, *Proc. Mach. Learn. Res.* **54**, 962–970 (2017).
26. P. S. Thomas, G. Theocharous, M. Ghavamzadeh, *Proc. Mach. Learn. Res.* **37**, 2380–2388 (2015).
27. M. Ghavamzadeh, M. Petrik, Y. Chow, *Adv. Neural Inform. Process. Syst.* **29**, 2298–2306 (2016).
28. R. Laroche, P. Trichelair, R. T. des Combes, *Proc. Mach. Learn. Res.* **97**, 3652–3661 (2019).
29. D. Bertsimas, V. Gupta, N. Kallus, *Math. Program.* **167**, 235–292 (2018).
30. M. Bastani, thesis, University of Alberta (2014).
31. S. Schmidt, K. Nørgaard, *J. Diabetes Sci. Technol.* **8**, 1035–1041 (2014).
32. C. Dalla Man *et al.*, *J. Diabetes Sci. Technol.* **8**, 26–34 (2014).
33. S. W. Suh, E. T. Gum, A. M. Hamby, P. H. Chan, R. A. Swanson, *J. Clin. Invest.* **117**, 910–918 (2007).
34. A. J. Bree, E. C. Puente, D. Daphna-Iken, S. J. Fisher, *Am. J. Physiol. Endocrinol. Metab.* **297**, E194–E201 (2009).
35. E. C. McNay, V. E. Cotero, *Physiol. Behav.* **100**, 234–238 (2010).
36. D. Precup, R. S. Sutton, S. Dasgupta, Off-policy temporal-difference learning with function approximation. In *Proceedings of the 18th International Conference on Machine Learning* (2001), pp. 417–424; https://dl.acm.org/citation.cfm?id=655817.
37. H. Zisser, L. Jovanovic, F. Doyle III, P. Ospina, C. Owens, *Diabetes Technol. Ther.* **7**, 48–57 (2005).
38. Data related to this publication are available through Harvard Dataverse. DOI: 10.7910/DVN/O35FW8
39. Source code for all experiments is available through Zenodo. DOI: 10.5281/zenodo.3490615
40. T. M. Mitchell, *Machine Learning* (McGraw-Hill, 1997).
41. D. E. Rumelhart, G. E. Hinton, R. J. Williams, *Nature* **323**, 533–536 (1986).
42. A. Liaw, M. Wiener, *R News* **2**, 18–22 (2002).

ACKNOWLEDGMENTS

We thank G. Theocharous and M. Ghavamzadeh for their guidance in the development of the high-confidence policy improvement algorithms that initiated this line of research, and multiple colleagues and reviewers who provided valuable feedback. **Funding:** Supported by a gift from Adobe, NSF

CAREER awards 1350984 (E.B.) and 1453474 (Y.B.), NSF grant 1763423, and Institute of Educational Science grant R305A130215. The opinions expressed are those of the authors and do not represent views of Adobe, NSF, the Institute, or the U.S. Department of Education. **Author contributions:** P.S.T. conceived the idea with A.G.B. providing early guidance. P.S.T. and E.B. developed the framework formalization. P.S.T., B.C.d.S., S.G., Y.B., and E.B. designed and executed the experiments and analyzed the data, with P.S.T. and B.C.d.S. focusing on the regression experiments, S.G. and Y.B. focusing on the classification experiments, and P.S.T. and E.B. focusing on the reinforcement learning experiments. P.S.T., B.C.d.S., A.G.B., S.G., Y.B., and E.B. wrote and edited the manuscript. **Competing interests:** The authors declare no competing interests. **Data and materials availability:** Data discussed in the main text and supplementary materials, as well as source code for reproducing all experiments, are available online (*38*, *39*).

SUPPLEMENTARY MATERIALS

science.sciencemag.org/content/366/6468/999/suppl/DC1
Supplementary Text
Figs. S1 to S39
References (*43*–*214*)

11 November 2016; resubmitted 31 August 2017
Accepted 25 October 2019
10.1126/science.aag3311

Preventing undesirable behavior of intelligent machines

Philip S. Thomas, Bruno Castro da Silva, Andrew G. Barto, Stephen Giguere, Yuriy Brun, and Emma Brunskill

Science **366** (6468), 999-1004.
DOI: 10.1126/science.aag3311

Making well-behaved algorithms

Machine learning algorithms are being used in an ever-increasing number of applications, and many of these applications affect quality of life. Yet such algorithms often exhibit undesirable behavior, from various types of bias to causing financial loss or delaying medical diagnoses. In standard machine learning approaches, the burden of avoiding this harmful behavior is placed on the user of the algorithm, who most often is not a computer scientist. Thomas *et al.* introduce a general framework for algorithm design in which this burden is shifted from the user to the designer of the algorithm. The researchers illustrate the benefits of their approach using examples in gender fairness and diabetes management.

Science, this issue p. 999

science.sciencemag.org/content/366/6468/999/suppl/DC1

Supplementary Materials for

Preventing undesirable behavior of intelligent machines

Philip S. Thomas*, Bruno Castro da Silva, Andrew G. Barto, Stephen Giguere,
Yuriy Brun, Emma Brunskill

*Corresponding author. Email: pthomas@cs.umass.edu

Published 22 November 2019, *Science* **366**, 999 (2019)
DOI: 10.1126/science.aag3311

This PDF file includes:

Contents

1 The Standard ML Approach for Designing Machine Learning Algorithms

When designing a machine learning algorithm using the current standard ML approach, the first step is to mathematically define what the algorithm should try to do—the goal of the algorithm. At an abstract level, this goal has the same form for almost all machine learning problems: find a *solution* $\theta^\star$, within some *feasible set* Θ, that maximizes an *objective function* $f : \Theta \to \mathbb{R}$. That is, the algorithm should search for an optimal solution

$$\theta^\star \in \arg\max_{\theta \in \Theta} f(\theta). \tag{S1}$$

A simple example is the design of an algorithm to solve a regression problem. Let X and Y be dependent real-valued random variables. The goal is to estimate Y given X. The first step is to specify the feasible set Θ to be a set of estimator functions chosen to model the relationship between X and Y. Each function $\theta \in \Theta$ takes a real number as input and produces a real number as output, that is, $\theta : \mathbb{R} \to \mathbb{R}$.

We might then define the objective function to be the negative *mean squared error* (MSE):

$$f(\theta) \coloneqq -\mathbf{E}\left[(\theta(X) - Y)^2\right]. \tag{S2}$$

This completes the specification of *what* the algorithm should do, and so one can begin working on *how* the algorithm should do it. For example, we might have the algorithm construct an estimate $\hat{f}$ of f using data consisting of m realizations of (X, Y), that is, (x_i, y_i) for $i = 1, \ldots, m$. For example, $\hat{f}$ could be the negative *sample MSE*:

$$\hat{f}(\theta) \coloneqq -\frac{1}{m}\sum_{i=1}^{m}(\theta(x_i) - y_i)^2, \tag{S3}$$

and the algorithm could return an element of $\arg\max_{\theta \in \Theta} \hat{f}(\theta)$.

Similarly, for a *classification problem*, Θ would be a set of classifiers and f a notion of how often the classifier selects the correct labels [43, 44, 45]. For an *unsupervised learning problem*, one might define Θ to be a set of statistical models and $f(\theta)$ to be a notion of how similar θ is to a target model [46].

The same steps are required to design an algorithm for a *reinforcement learning problem*. In this case, we might define Θ to be a set of policies (functions that map states to probability distributions over actions) and f to be an objective function such as the expected discounted return [47]. We might then design an algorithm that searches for a solution θ that maximizes an estimate $\hat{f}(\theta)$ of $f(\theta)$ constructed from a sample of state-action-reward trajectories, or we might design an algorithm like Q-learning [48] that indirectly maximizes f by optimizing a related function.

2 Limitations of the Standard Approach

Once a machine learning expert has designed a machine learning algorithm, the algorithm can be used as a component of a larger system, which we will call the *agent*, that uses one or more machine learning algorithms for a particular application, or range of applications. We

call the person designing the agent the *user* of the machine learning algorithm. The user can be nearly anyone, from a child working with LEGO Mindstorms, to a businessperson using Microsoft Excel to fit a line to data points, to a (non-computer) scientist using data analysis tools to analyze research data, to a machine learning researcher using reinforcement learning for part of a controller of an autonomous vehicle. The primary limitation of the standard ML approach to designing a machine learning algorithm is that it does not make it easy for an algorithm's user (who may or may not be a machine learning expert) to specify and regulate the desirable and undesirable behavior of the agent. Although in principle there might be definitions of Θ and f that prevent the algorithm from leading to undesirable behavior, in practice there is no way for the user to know these definitions without performing additional data analysis that can be challenging even for a machine learning expert.

As an example, consider a reinforcement learning application where an artificial neural network is used to control a robot. In this context, the feasible set Θ is a set of neural networks (typically with the same structure, but different weights), each of which would cause the robot to behave differently. Reinforcement learning algorithms can be used to search for an artificial neural network within Θ that performs well in terms of some user-defined performance measure. The user of a reinforcement learning algorithm might want to implement Asimov's first law of robotics, which essentially states that a robot may not harm a human [49]. However, the user of the algorithm typically does not know whether any particular artificial neural network θ will cause the robot to harm a human or not. This means that the user of the algorithm cannot specify Θ to only include artificial neural networks that produce safe behavior. Additionally, since most reinforcement learning algorithms are not guaranteed to produce an optimal solution given finite data, simply adding a penalty into the objective function to punish harming a human does not preclude the algorithm from returning a suboptimal solution (artificial neural network) that causes harm to a human. This means that the user of the algorithm has no easy way to constrain the behavior of the agent—the user of the algorithm must have deep knowledge of the environment that the robot will be faced with, what each artificial neural network $\theta \in \Theta$ does, and how the reinforcement learning algorithm works, to ensure that the reinforcement learning algorithm will not cause the robot (agent) to harm a human.

2.1 An Example

As an illustrative example we consider a problem for which it is difficult to prevent undesirable behavior. The example concerns the *fairness* of one of the most widely used and well-studied data analysis tools: linear regression. While fairness of machine learning algorithms is an important topic of considerable current interest and research [50, 3, 51], we emphasize that the new approach we are proposing is not limited to addressing this issue. Here, fairness (or lack thereof) merely provides an accessible example of an algorithm's misbehavior that our approach is designed to mitigate.

The goal in our illustrative example is to predict the aptitudes of job applicants based on numerical values describing the qualities of their résumés. Although later we consider a similar problem using advanced regression algorithms and actual data, here we use synthetic data and only consider linear regression algorithms. We show how linear regression algorithms designed using the standard ML approach can result in predictions of applicant performance

that systematically discriminate against a group of people (such as people of one gender or race).

Let each applicant be in one of two sets, $\mathcal{A}$ and $\mathcal{B}$. For example, $\mathcal{A}$ could be the set of all possible female applicants and $\mathcal{B}$ could be the set of all possible male applicants.[1] We call the group that the applicant belongs to his or her *type*. We are given a training set that contains data describing the résumés of $m = 1{,}000$ previous applicants, the applicants' actual aptitudes (as determined by their observed performances), and their types. For each $i \in \{1, \ldots, m\}$, let $x_i \in \mathbb{R}$ be a number describing the quality of the i^{th} applicant's résumé, let $y_i \in \mathbb{R}$ be a measure of the applicant's actual aptitude (which we would like to predict given x_i), and let $t_i \in \{0, 1\}$ be an indicator of the applicant's type: $t_i = 0$ if the applicant is in $\mathcal{A}$ and $t_i = 1$ if the applicant is in $\mathcal{B}$. For simplicity, we assume that all terms, x_i and y_i, have been normalized so that they each are in the interval $[-3, 3]$.

We are tasked with using the training set to find a linear estimator (an affine function) $\hat{y}(x, \theta) \coloneqq \theta_1 x + \theta_2$ that predicts y given x, where x is the number describing the quality of a new applicant's résumé, y is the unknown aptitude of this new applicant, and the vector $\theta \coloneqq [\theta_1, \theta_2]^\intercal$ is a *weight vector* in the feasible set $\Theta = \mathbb{R}^2$. Although each applicant's type is available in the training set, we do not assume that we will know a new applicant's type: résumés do not typically include applicants' genders or ethnicities. If our estimator is used to filter actual résumés submitted to a company so that only a subset of applications receive human review, as described by Weber [16] and Miller [52], then to comply with anti-discrimination laws, we might want to ensure that our estimator does not produce racist or sexist behavior, meaning that it does not discriminate against people in $\mathcal{A}$ or $\mathcal{B}$.

Of many ways that we might choose to define undesirable discrimination behavior, we choose one for illustrative purposes. We do not suggest that this definition captures all possible types of discrimination. It is rather one example of behavior that cannot easily be precluded when using algorithms designed using the standard ML approach to designing machine learning algorithms. Intuitively, we define *undesirable discrimination behavior* to occur when the algorithm produces predictions that are, on average, too high for people of one type and too low for people of the other type. Importantly, we do not consider it to be undesirable discriminatory behavior if the algorithm produces larger predictions, on average, for people of one type, since people of one type might actually have higher aptitudes on average. However, if the algorithm's result over-predicts the performance of men by 10% and under-predicts the performance of women by 10%, on average, then we would say that the algorithm discriminates against women.

To formalize this notion, we define the following *discrimination statistic*, $d(\theta)$:

$$d(\theta) \coloneqq \underbrace{\mathbf{E}\Big[u\left(\hat{y}(X, \theta) - Y\right) \Big| T = 0\Big]}_{\text{(a)}} - \underbrace{\mathbf{E}\Big[u\left(\hat{y}(X, \theta) - Y\right) \Big| T = 1\Big]}_{\text{(b)}}, \tag{S4}$$

where X, Y, and T are random variables that denote the numerical measure of résumé quality, actual aptitude, and applicant type, $u : \mathbb{R} \to \mathbb{R}$ is a *utility function*, and (a) and (b) respectively indicate how much the estimator over-predicts on average for people in $\mathcal{A}$ and $\mathcal{B}$. The utility function, u, determines the relative importance of different amounts of over- and under-prediction. For simplicity here we assume that u is the identity function.

[1] For simplicity and to match the data that we use later, we consider the simplified binary-gender setting.

Given a training set that contains data from 1,000 past applicants, we would like to apply a linear regression algorithm to get accurate predictions of applicant aptitudes while simultaneously ensuring that the absolute value of the discrimination statistic, $|d(\theta)|$, is small. Fig. S1 shows an example training set along with the linear estimator produced by least squares linear regression.

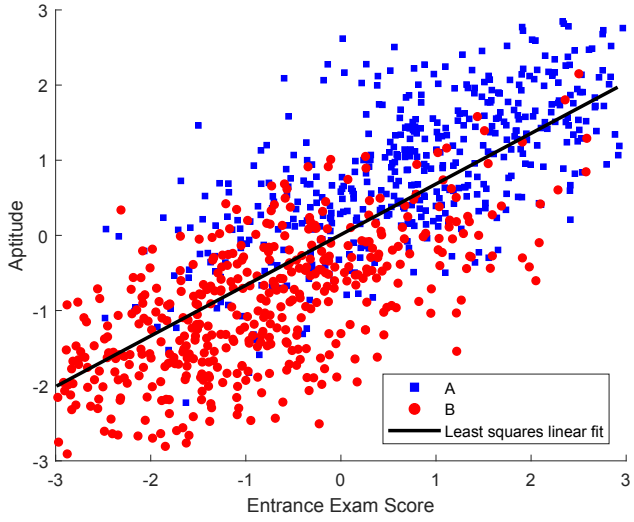


Fig. S1. An example of a training set from the illustrative example. This figure depicts the information that the user is given when applying a linear regression algorithm. The user does not know the true underlying distribution of the data, but rather only has these $m = 1{,}000$ samples. The black line is the linear estimator produced by least squares linear regression.

Since the linear regression algorithm's objective is to make accurate predictions, we might expect it to be impartial as to whether people are in $\mathcal{A}$ or $\mathcal{B}$, and so it should not tend to produce discriminatory behavior: its impartiality should make it fair to all people. However, this is not always the case, as we can illustrate using our example. Suppose that the problem has the following properties, which are unknown to the algorithm's user: future applicants are in $\mathcal{A}$ and $\mathcal{B}$ with equal probability, and the training set has an equal number of applicants of each type; $Y \sim \mathcal{N}(1, 1)$ if $T = 0$ and $Y \sim \mathcal{N}(-1, 1)$ if $T = 1$, where $\mathcal{N}(\mu, \sigma^2)$ denotes the normal distribution with mean μ and standard deviation σ; $X \sim \mathcal{N}(Y, 1)$ (an applicant's résumé quality is equal to their true aptitude, plus random noise with a standard normal distribution); the training set, $D = \{(X_i, Y_i, T_i)\}_{i=1}^{m}$, contains $m = 1{,}000$ realizations of X, Y, and T.

Given these properties, we generated 10,000 independent training sets, each containing data from $m = 1{,}000$ applicants. We computed the least squares linear fits for each data set and computed the true discrimination statistic, $d(\theta)$, for each of the resulting linear estimators using our (but not the user's) knowledge of the true distribution of the data. The mean discrimination statistic was -0.67, which is a large amount of discrimination in favor of people in group $\mathcal{B}$, given that applicant aptitudes tend to be in the range $[-3, 3]$. Fig. S2 shows the least squares linear fits for all of the 10,000 trials.

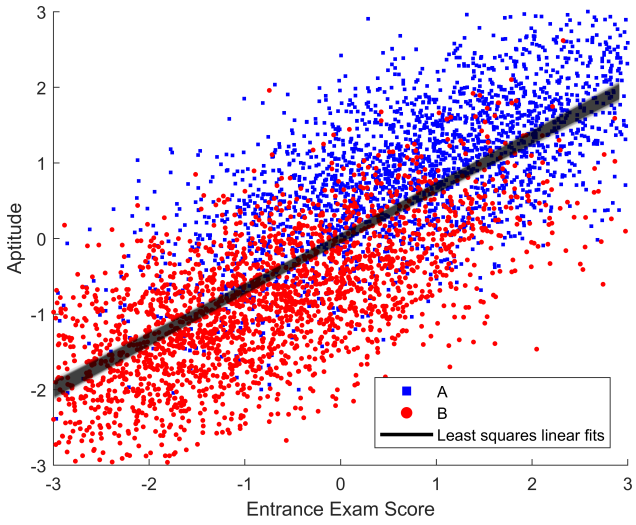


Fig. S2. Linear fits from least squares regression for 10,000 trials using transparent lines to show the distribution of the linear estimators. The background contains 5,000 realizations of (X, Y, T) to give an idea about the true underlying distribution of the data. The lines tend to over-predict for red points (people of type $T = 1$) and under-predict for blue points (people of type $T = 0$).

2.2 Potential Remedies

Computers do not have an inherent desire to produce undesirable behavior (for example, to harm humans or to be racist or sexist) and the user of an algorithm typically will not provide the algorithm with direct incentives to produce undesirable behavior. It is therefore tempting to rely on the impartial nature of the machine learning algorithm—the fact that it was not instructed to produce undesirable behavior. However, as our example illustrates for the case of discriminatory behavior of linear regression, a lack of direct incentives to produce undesirable behavior does not preclude undesirable behavior. Since we cannot rely on a machine learning algorithm designed using the standard ML approach to avoid undesirable behavior, a number of approaches can be taken in an attempt to remedy this problem. Before presenting our new approach, we describe a collection of possible cures, highlighting their shortcomings that our new approach avoids.

2.2.1 Determine and Combat the Root Causes of Undesirable Behavior

When undesirable behavior occurs, it is natural to wonder what the root cause of the behavior was. Was the undesirable behavior caused by improper use of a machine learning algorithm? Would a different way of using an algorithm designed using the standard ML approach preclude the undesirable behavior? In general: what could have been done differently by the user of the algorithm to cause the agent to not produce the undesirable behavior? This is the question that we do not want the user of an algorithm to have to answer, since it requires detailed knowledge of the problem to answer and is prone to being answered incorrectly. At a high level, our argument in favor of our new approach is that this process places an undue burden on the user of a machine learning algorithm, a user who may not be a machine

learning expert, and this burden should be shifted away from the user.

The severity of this burden is obvious for complicated problems: the user of a complicated reinforcement learning algorithm that tunes the weights in a neural network controlling a robot cannot be expected to know which weights or settings of the algorithm would result in the robot eventually harming a human. Perhaps less obviously, this problem impacts even simple data analysis algorithms, as illustrated by our linear regression example. What would be required to understand and mitigate the root causes of undesirable discriminatory behavior?

One possible cause of discriminatory behavior might be an imbalance in the training data. If more applicants from group $\mathcal{B}$ are observed than applicants from $\mathcal{A}$, then we might expect the regression algorithm to favor solutions that produce more accurate predictions for people in $\mathcal{B}$, even if it results in worse predictions for people in $\mathcal{A}$. However, this is not the cause of discriminatory behavior in our example because we defined the underlying applicant distribution so that there is no minority group.

Another possible cause of discriminatory behavior might be bias in the data set. If the training data set was biased so that it over-reported the aptitudes of applicants in $\mathcal{A}$ relative to their true aptitude, then the regression algorithm would also be biased towards discriminating in favor of applicants in $\mathcal{A}$. However, there is no additional bias in the training data of our example because the training and testing data come from the same distribution.

Perhaps the choice of a linear estimator is at fault. The least squares estimator may not be linear, and the closest linear approximation happens to discriminate. This suggests that using a more appropriate class of estimators might mitigate undesirable discriminatory behavior. This is straightforward to test in our example because we know the true underlying distribution of applicants. In the appendix of the supplementary materials we show that, for any résumé quality, x, the least squares estimate of aptitude (not just the *sample* least squares estimate) is $\hat{y}(x) = \frac{2}{3}x$, given the distributions we assumed. That is, the least squares estimates for each possible résumé quality x is given by a linear function, and so the optimal estimator (in terms of MSE) is in our chosen function class.

Another possible cause of discriminatory behavior might be the use of finite data sets. Since the training set is finite and generated randomly, different data sets will result in different linear fits. Perhaps all the linear fits are centered around an estimator that does not discriminate, but the way that they vary about this estimator causes the absolute value of the mean discrimination statistic to be large. Because we know the underlying distribution of applicants, we can check this by solving for the single linear estimator that would be produced if given an infinite amount of data. Because the least squares estimator is linear, as discussed previously, with infinite data we would obtain the estimator $\hat{y}(x) = \frac{2}{3}x$ (with probability 1). Using our knowledge of the underlying distribution of applicants again, the discrimination statistic $d(\theta)$ for this estimator is -0.67. Thus, even if there were an infinite amount of data, the linear least squares regression algorithm still discriminates.

Another possible cause of discriminatory behavior could stem from the decision of whether the regression algorithm's predictions can depend on the applicant's type: whether the applicant is in $\mathcal{A}$ or $\mathcal{B}$. One might think that an algorithm could not possibly discriminate if it has no access to the applicants' types, but this is not the case. The linear regression algorithm in our illustrative example is blind to the type of each applicant, but it still discriminates. Alternatively, one might expect the opposite: that if applicants' aptitudes depend on their

types, then the algorithm should have access of the applicants' types so that it can make fair predictions. For example, we could apply the least squares regression algorithm twice, once to data on applicants in $\mathcal{A}$, and once to data from applicants in $\mathcal{B}$. We could then use the $\mathcal{A}$ estimator to predict the aptitude of a new applicant if they are in $\mathcal{A}$, and the $\mathcal{B}$ estimator otherwise.

Three conditions must be met for this approach to be effective in precluding discrimination. First, the regression algorithm must have access to new applicants' types, which may not be the case. Résumés likely will not include genders or ethnicities. Second, sufficient data must be available. If the available data is insufficient, then this approach might produce discriminatory behavior because the training data could be an unlikely sample that does not reflect the actual distributions of X, Y, and T. Determining the amount of data necessary for this scheme to limit discrimination with high probability would require the user to perform additional data analysis. Furthermore, one might conjecture that with small amounts of data, the expected value of the discrimination statistic will be close to zero. However, this is not always the case. Fig. S3 provides an example using a distribution of applicants that is different from the distribution used above. Regression with small amounts of data can still cause the expected value of the discrimination statistic to be large. The third requirement for mitigating discriminatory behavior using knowledge of applicants' types is that the utility function must be the identity function. For every linear regression problem the estimator that minimizes the MSE (the actual MSE, not the sample MSE) causes the mean of the unsquared error to also be zero, and so this scheme results in a discrimination statistic close to zero given enough data. However, the estimator that minimizes the MSE does not necessarily cause the mean *utility* of the error to also be zero, and so utility functions other than the identity function can result in this scheme producing discriminatory estimators even given infinite data.

It should now be clear that the root cause of the discriminatory behavior is not obvious and can easily be overlooked or incorrectly attributed. This is particularly true if the user of the algorithm is not trained in data analysis methods. In our example, the actual root cause of discriminatory behavior when using ordinary least squares linear regression arises from the fact that the objective function calls for minimizing MSE, which is at odds with minimizing the discrimination statistic. To minimize the discrimination statistic, ideally to make it zero, requires the machine learning algorithm to return an estimator that does not minimize the MSE. The estimator that minimizes the (sample) MSE can still have mean errors that are different for people of different types. In this example the mean error of the least squares fit tends to be positive for people with low aptitudes and negative for people with high aptitudes. Because people in $\mathcal{B}$ tend to have lower aptitudes, minimizing the MSE results in discrimination in favor of people in $\mathcal{B}$.

In summary, although it might be possible to determine and combat the root causes of undesirable behavior, doing so can be difficult, error prone, and can require data analysis, even for simple well-understood algorithms like linear regression. However, so far we have discussed algorithms that were not designed with an explicit goal of allowing the user to control their behavior. We now review variants of algorithms designed using the standard ML approach that are intended to provide guarantees about the behavior of agents. Below we argue that our new approach addresses many of the shortcomings of these approaches to remedying a machine learning algorithm's undesirable behavior.

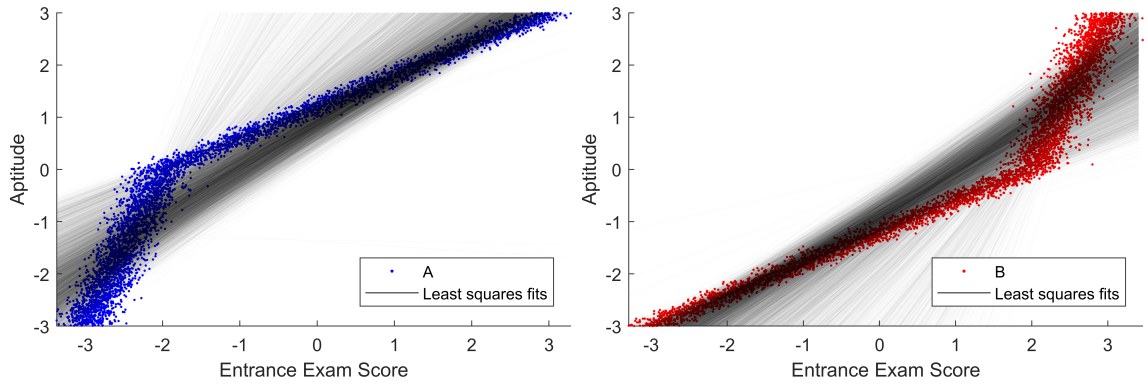


Fig. S3. Different distributions for people of types $\mathcal{A}$ (left) and $\mathcal{B}$ (right), such that using independent linear regressors for people of each type, and training data from five people of each type, results in an average discrimination statistic over 1,000 trials of 0.42.

2.2.2 Potential Remedy: Hard Constraints

Some optimization algorithms allow the user to place constraints on Θ, such as the simplex algorithm for linear programs, which requires the user to define the feasible set using linear constraints [53]. Thomas et al. [54] and Le et al. [55] propose reinforcement learning algorithms that allow the user to specify a "safe" set of policies, and guarantee that the algorithms will always converge to policies contained in this safe set. However, these authors sidestep the question of how this safe set of policies can be determined by assuming that it is provided a priori. Determining which solutions are "safe"—which solutions do not result in undesirable behavior—can be difficult, requires detailed knowledge of the problem at hand, and can sometimes be impossible. Although some work has considered how hard constraints can easily be specified by a user, e.g., by providing examples of desirable and undesirable behavior [56], these approaches do not provide practical guarantees about the quality of the constraints that they produce.

Consider again our illustrative example. Without constraints, the feasible set is the set of all possible pairs of weights that define a line: $\Theta = \mathbb{R}^2$. We might wish to define the set of safe solutions $\mathcal{S} \subseteq \mathbb{R}$ to be all of the solutions that result in discrimination statistics with magnitude at most some small value ϵ. That is, $\mathcal{S} \coloneqq \{\theta \in \mathbb{R}^2 : |d(\theta)| \leq \epsilon\}$. The resulting least squares algorithm could then be constrained to only consider solutions in $\mathcal{S}$. The problem here is that we cannot know $\mathcal{S}$ without knowledge of the underlying distribution of applicants: their types, aptitudes, and résumé qualities.

Nevertheless, constraints on the feasible set can be an effective means for precluding some specific definitions of undesirable behavior, provided that the user has access to detailed knowledge of the problem. For example, control theoretic algorithms have been developed that ensure (sometimes with high probability, rather than surely) that a system will never enter a predefined unsafe set of states [57, 58, 59, 60, 61, 62, 63, 64].

2.2.3 Potential Remedy: Soft Constraints

Rather than constrain Θ, we might consider modifying f (or an estimate $\hat{f}$ thereof) so that it penalizes undesirable behavior. These sorts of penalties in the objective function are

sometimes called *soft constraints* [9] because the algorithm is driven to satisfy them but has the freedom to violate them to obtain a large improvement according to the original objective function. Soft constraints are also sometimes called *penalty functions* and are related to *barrier functions* [65], which penalize solutions that are close to the boundary of a feasible set.

Although soft constraints can sometimes be effective, they have a significant drawback: they require a parameter $\lambda \in \mathbb{R}_{\geq 0}$ that scales the importance of the soft constraint relative to the primary objective. If λ is set too far to one extreme, the algorithm will ignore the soft constraint and focus entirely on the primary objective, which means that undesirable behavior could result. If λ is set too far to the other extreme, the algorithm will ignore the primary objective and focus solely on ensuring that undesirable behavior does not occur. Properly selecting λ is not simple and requires additional data analysis that may be unreasonable to expect from a user not trained in data analysis.

Again consider our illustrative example. Here we might introduce a soft constraint so that the objective calls for the simultaneous minimization of the MSE and the absolute value of the discrimination statistic. That is, if $\Theta = \mathbb{R}^2$ so that each $\theta \in \Theta$ is a possible weight vector defining a line, then:

$$f(\theta) \coloneqq -\operatorname{MSE}(\theta) - \lambda |d(\theta)| \tag{S5}$$

$$= -\mathbf{E}\left[(\hat{y}(X,\theta) - Y)^2\right] - \lambda \left| \mathbf{E}\Big[\hat{y}(X,\theta) - Y \Big| T = 0\Big] - \mathbf{E}\Big[\hat{y}(X,\theta) - Y \Big| T = 1\Big] \right|. \tag{S6}$$

We might then define our data-based estimate of $f(\theta)$ to use the sample MSE and sample discrimination statistic:

$$\hat{f}(\theta) \coloneqq -\frac{1}{m}\sum_{i=1}^{m}(\hat{y}(x_i,\theta) - y_i)^2 - \lambda \left| \left(\frac{1}{\sum_{i=1}^{m}(1-t_i)} \sum_{i=1}^{m} (1-t_i)\left(\hat{y}(x_i,\theta) - y_i\right) \right) - \left(\frac{1}{\sum_{i=1}^{m} t_i} \sum_{i=1}^{m} t_i \left(\hat{y}(x_i,\theta) - y_i\right) \right) \right|. \tag{S7}$$

Although in some cases it is reasonable to expect the user of a machine learning algorithm to be able to specify a soft constraint of this form, it is typically not reasonable to expect the user to be able to select the value of λ properly. As an example, Fig. S4 depicts the mean discrimination statistic and least squares solutions $\hat{\theta}^\star \in \arg\max_{\theta\in\Theta} \hat{f}(\theta)$ when using Eq. S7 with various choices of λ. As λ increases, MSE increases as well, since the objective function places increasing weight on avoiding discrimination at the cost of the error. The left plot of Fig. S4 depicts the absolute discrimination statistic of the solutions produced using various λ. As λ increases, the absolute value of the discrimination statistic decreases, as expected.

To ensure that the expected absolute value of the discrimination statistic is at most some value ϵ, we can use the plot on the left to find the smallest value of λ that produces discrimination statistics that are less than ϵ on average. For $\epsilon = 0.1$, this is $\lambda \approx 4.9$. However, in practice these plots are not available to the user. They were generated by additional data analysis, which required using large amounts of data that would not generally be available in practice.

Ideally, the machine learning algorithm should use the available data to automatically optimize the value of λ in such a manner that, when combined with finite sample bounds,

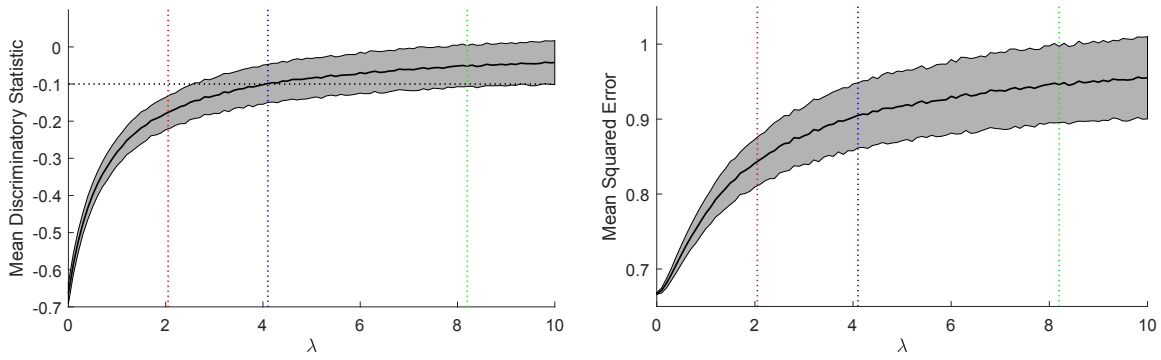


Fig. S4. Mean discrimination statistic (left) and MSE (right) produced by the solutions found using various values of λ to specify the importance of a soft constraint. The dotted blue line is the smallest value of λ for which the absolute value of the mean discrimination statistic is less than 0.1 (that is, $\lambda \approx 4.9$), and the red and green line are half and double this value. Both plots are averaged over 1,000 trials (each from a new sampling of the training set) and the upper and lower standard deviation intervals are shaded. We show standard deviation because standard error bars are too small to be clearly visible.

provides the user with confidence that the algorithm will not cause discriminatory behavior. Such an algorithm would be an instance of the type of algorithm that we propose. It shifts from the user of the algorithm to the algorithm itself the burden of understanding the given problem well enough to adequately decrease the probability of undesirable behavior.

2.2.4 Potential Remedy: Multiple Objectives

The addition of soft constraints is one way to express the idea that the true objective is multifaceted: the agent should optimize a primary objective while also optimizing other objectives that measure the prevalence of undesirable behavior. Multiobjective optimization algorithms are optimization algorithms designed specifically for problems with multiple, and typically conflicting, objectives. Modern multiobjective optimization algorithms are usually based on the concept of the *Pareto frontier*. A solution is on the Pareto frontier if there does not exist another solution that causes any of the objectives to increase without decreasing at least one of the other objectives (assuming that larger values are better for all objectives). Solutions on the Pareto frontier provide a balance between the different objectives, and an algorithm should ideally return a solution on the Pareto frontier since any other solution could be improved with respect to at least one objective function without hindering performance with respect to any of the other objective functions.

While algorithms exist to compute the Pareto frontier for a wide variety of multiobjective machine learning problems [66, 67, 68], knowing the Pareto frontier does not provide a complete solution. One must still decide which solution from the Pareto frontier to use. The user must still explicitly select a trade-off between the different objective functions. In our illustrative example we might use two objective functions: the negative MSE of the estimator, $-\text{MSE}(\theta)$, and the negative absolute value of the discrimination statistic, $-|d(\theta)|$. Each estimator that results from using a soft constraint with any value of λ is an element of the Pareto frontier, and so the user must still effectively determine how to select λ.

Below we present two linear regression algorithms, NDLR and QNDLR, that were designed using our framework. Although these algorithms do not have a λ parameter, they do have a different parameter, ϵ. Specifically, they guarantee that with high probability the absolute value of the discrimination statistic will be no larger than ϵ. The difference between requiring the user to select λ and requiring the user to select ϵ is subtle but important. The scale (or unit) associated with ϵ is one that can be understood by the user without any problem-specific data analysis. For the discrimination statistic Eq. S4, ϵ is the maximum difference in mean prediction errors, and so its associated scale is the scale of errors. By contrast, the scale (or unit) associated with λ depends on the MSE of the solution, which typically is not known to the user, and the estimation of which requires additional data analysis. Thus, we contend that it is easier for the user to select ϵ than it is for the user to select λ.

Furthermore, regardless of λ's value, using a soft constraint can still result in solutions that discriminate significantly. This is due to the random nature of data. Small data sets may not accurately represent the true underlying distribution of data. As a result, a soft-constrained algorithm may select a solution that has a *sample* discrimination statistic of zero on the available data, but which actually has a large discrimination statistic (discriminates when presented with new data). This behavior is evident in our later experiments (cf. Fig. S16b), in which soft-constrained methods using small amounts of data and large values of λ produce solutions that discriminate significantly. By contrast, with high probability, the algorithms that we design using our framework do not discriminate regardless of how much data they are presented with. This is because they incorporate mechanisms that automatically determine whether the random nature of data is causing them to draw false conclusions.

This is not to say that our framework for designing machine learning algorithms should replace multiobjective methods. On the contrary, our framework and multiobjective methods (1) have different use cases and (2) can be used in conjunction with one another for problems that satisfy both use cases. Our framework is intended for applications where there is at least one constraint that is strictly of more importance than others (avoiding undesirable behavior), but which also cannot be satisfied with certainty. This is particularly true for applications in which safety is a paramount concern: subject to the constraint that the algorithm ensures with high probability that its behavior is safe, it is free to optimize some objective (or multiple objectives). Multiobjective methods, on the other hand, are primarily suited to applications where there is a trade-off between multiple objectives—there is no one objective that is of paramount importance.

To make this difference clear, consider an example that satisfies the use cases of both approaches: optimizing treatment policies for type 1 diabetes. Later we describe this application in more detail—here we provide an overview. The goal in this application is to find treatment policies (which define how much insulin a person with type 1 diabetes should inject prior to eating a meal) that keep a person's blood glucose near optimal levels. Thus, the objective function gives a measurement of how much the person's blood glucose deviates from optimal throughout a day. There is also an important safety constraint: some treatment policies can result in a condition called *hypoglycemia* that drastically increases the probability that a person will die within five years [69]. However, hypoglycemia cannot be avoided with certainty. Thus, if we apply a machine learning algorithm to automatically improve a treatment policy proposed by a physician, we may wish to include a safety constraint: our algorithm should guarantee with high probability that it will not change the person's

treatment policy to one that increases the prevalence of low blood glucose relative to the initial treatment policy specified by the physician. This safety constraint should be inalienable—it should not be sacrificed or traded-off with the objective function.

However, this problem also has a multiobjective component: we should favor treatment policies that inject less insulin, as there are negative effects associated with long-term elevated levels of insulin. These effects are minor in comparison to the risks associated with hypoglycemia, and so we can treat this as a multiobjective problem wherein our goal is to simultaneously minimize the amount of insulin that is injected and keep blood glucose near optimal levels. Thus this problem has both a safety component that can be handled using our framework (precluding increases in the prevalence of low blood glucose), and a multiobjective component (trading off the competing objectives of maintaining optimal blood glucose levels and injecting as little insulin as possible).

2.2.5 Potential Remedy: Chance-Constraints

Many applications may not have any solutions that preclude undesirable behavior with certainty. Requiring undesirable behavior to never occur is too strict in these cases, and so weaker requirements are called for. This is what the *chance-constrained program* formulation does by allowing constraints on the probability that a solution will result in undesirable behavior. Chance-constrained programs were pioneered by Charnes and Cooper [11], Miller and Wagner [70], and Prékopa [71], and can be expressed formally as:

$$\arg\min_{\theta \in \Theta} f(\theta) \tag{S8}$$

$$\text{s.t. } \forall i \in \{1, \ldots, n\},\ \Pr(g_i(\theta, W_i) \leq 0) \geq 1 - \delta_i, \tag{S9}$$

where $f : \Theta \to \mathbb{R}$, each g_i is a deterministic real-valued function, each W_i is a random variable, and each $\delta_i \in (0, 1)$. Crucially, not only are Θ, g_i, and δ_i all known, but the distribution of each W_i is known. Also, typically f is a convex function and Θ is a convex set. Chance-constrained programs have been extensively studied within the machine learning and optimization communities [72, 73], often with the expressed intent of better controlling agent behavior [74].

As an example of when chance-constraints might be appropriate, consider the search for a neural network $\theta \in \Theta$ for controlling a robot. One can think of $\mathcal{W}$ as the set of possible environments (or *worlds*) in which the robot could find itself in the future, and $W_i \in \mathcal{W}$ is a random variable that denotes the actual environment that the robot will be faced with. There may not exist a neural network $\theta \in \Theta$ that guarantees with certainty that the robot will never harm a human regardless of which environment $W_i \in \mathcal{W}$ it faces. Chance constraints allow one to define a "safe" set of solutions $\mathcal{S} \subseteq \Theta$ such that each $\theta \in \mathcal{S}$ ensures that with high probability the robot will not harm a human in the future, given that the future environment W_i is drawn from some known distribution. That is, $\mathcal{S} = \{\theta \in \Theta : \forall i \in \{1, \ldots, n\}, \Pr(g_i(\theta, W_i) \leq 0) \geq 1 - \delta_i\}$, where $g_i(\theta, w) > 0$ denotes that, if the future environment is $w \in \mathcal{W}$, then the solution θ will result in harm to a human.

Although chance-constrained programs can be useful for many real problems [75, 76, 77], like approaches based on hard constraints on the feasible set, existing chance-constrained algorithms are mostly limited in applicability to problems in which the user has significant

knowledge about the given problem. Specifically, the majority of such algorithms are a viable problem formulation when the user knows the distribution of each W_i. However, for many problems these distributions will be unknown. In our illustrative example, the random variable W_i might denote a single point (X, Y, T), it might denote a set of points (such as the entire training set), or it might denote the joint distribution over (X, Y, T) if we assume that the distribution of applicants is itself sampled from some meta-distribution. In all of these cases, the user does not have access to the distribution of each W_i, and therefore cannot construct $\mathcal{S}$.

Several variants of chance constrained programs weaken the assumption that the distribution of each W_i is known. For example, *ambiguous chance constrained programs* allow the distribution, P, of W_i to be unknown, as long as it is within some set of possible distributions $\mathcal{P}$ that is known [78]. Ambiguous chance constrained programs then require the chance constraints to hold for all $P \in \mathcal{P}$. *Scenario approximation* methods can apply when P is neither known nor within a known set, but samples (realizations of W_i) can be generated from P [79, 80, 81]. In general, *stochastic programming* methods allow for many variants of optimization problems wherein there is uncertainty about some of the parameters of the optimization problem [82]. However, these variants tend to include strong assumptions about the form of the problem: primarily that the objective function and feasible set are convex, and that the distributions of random variables are restricted to a specific class (e.g., Gaussians). To the best of our knowledge, none of the variants are able to handle our illustrative example. They do not allow for the specification that the objective is to minimize the MSE of the linear estimator, subject to the constraint that with high probability the absolute discrimination statistic $|d(\theta)|$ of the returned solution θ is bounded by some small constant ϵ. We discuss data-driven constrained optimization approaches in Section 6.

2.2.6 Potential Remedy: Robust or Risk-Sensitive Methods

Determining which solutions will produce undesirable behavior is difficult because typically there is uncertainty about the environment with which an agent will interact. *Robust optimization* algorithms address uncertainty about the environment by favoring solutions that work reasonably well across all of the possible environments, in contrast to seeking a solution that is expected to be optimal for one particular environment [12]. Robust optimization algorithms have been proposed as a means for ensuring that machine learning algorithms, ranging from supervised learning algorithms [83] to reinforcement learning algorithms [84], are in some way "safe" to use. However, there are several reasons that robust optimization is not a remedy to the issues of concern to us. Robust methods still require the algorithm's user to have detailed knowledge of the given problem to preclude undesirable behavior. Most robust optimization algorithms perform poorly if the user is unable to define a small set of environments that contains the actual environment with high probability.

Similarly, *risk-sensitive* optimization methods (which are sometimes viewed as types of robust optimization methods) use an objective function f that captures different measures of the distribution of outcomes that can result from using a solution θ. For example, if each $\theta \in \Theta$ defines a different controller for a robot, then $f(\theta)$ is often chosen to be a measure of the *expected* performance of the robot if it uses the controller θ. Risk-sensitive methods might define $f(\theta)$ to also penalize solutions that cause the observed performance of the robot

to have high variance [85]. Alternatively, risk-sensitive methods might define $f(\theta)$ to be the *expected shortfall* or *conditional value at risk* (CVaR) of the solution θ, a measure of the expected performance of the robot during the worst trials when using controller θ [86, 87]. Although applying risk-sensitive methods typically does not require detailed problem-specific knowledge, these algorithms do not allow the user to specify undesirable behavior from within a broad class of possible behavior. Furthermore, standard risk-sensitive methods typically do not provide practical guarantees about the performances of the solutions that they return.

3 A New Framework for Designing Machine Learning Algorithms

So far we have discussed the standard ML approach for designing machine learning algorithms, highlighting its limitations. We now turn to defining a framework for designing machine learning algorithms that allows the user to define and regulate desirable and undesirable behavior without the need for detailed knowledge of a given problem. The first step in our framework is not to define the goals of the *algorithm*, but instead to define the goals of the *designer* of the algorithm. Thus, the crux of our framework is a new mathematical problem formalization, which we call a *Seldonian optimization problem* (SOP) as an homage to Isaac Asimov's fictional character, Hari Seldon, a resident of a universe where Asimov's three laws of robotics failed to adequately control agent behavior due to their non-probabilistic requirements, and who formulated and solved a machine learning problem that would likely have required probabilistic constraints [13]. Before defining an SOP formally, we discuss the shortcomings of two potential alternatives that are flawed, but which might at first seem more reasonable.

3.1 Potential New Approach: Place Constraints on the Probability that a Solution is Safe

One would like to specify that an algorithm should guarantee, with high probability, that undesirable behavior will not result from its use. Let $\mathcal{S} \subseteq \Theta$ be the set of *safe solutions*. These are the solutions that do not cause undesirable behavior or that cause the probability of undesirable behavior to be sufficiently small. One might provide the user with a language for specifying $\mathcal{S}$ and then use a problem formulation of the form:

$$\theta^\star \in \arg\max_{\theta \in \Theta} f(\theta) \tag{S10}$$

$$\text{s.t. } \Pr(\theta \in \mathcal{S}) \geq 1 - \delta, \tag{S11}$$

where $\delta \in [0, 1]$ is a user-specified confidence level. The problem with this approach is that it is not meaningful to reason about the probability that a particular solution, θ, is in $\mathcal{S}$. Either $\theta \in \mathcal{S}$ or $\theta \notin \mathcal{S}$, and so $\Pr(\theta \in \mathcal{S})$ is necessarily either 0 or 1, and the above problem formulation is equivalent to

$$\theta^\star \in \arg\max_{\theta \in \mathcal{S}} f(\theta), \tag{S12}$$

which suffers from the problem that we do not know $\mathcal{S}$ a priori.

3.2 Potential New Approach: Place Constraints on the Agent's Data-Driven Belief that a Solution is Safe

One way to avoid the problem described in the previous section is to include constraints that are based on the agent's data-driven beliefs about whether $\theta \in \mathcal{S}$, rather than the probability that $\theta \in \mathcal{S}$. That is, one might use a problem formulation of the form:

$$\theta^\star \in \arg\max_{\theta \in \Theta} f(\theta) \tag{S13}$$

$$\text{s.t. } \mathcal{B}(\theta \in \mathcal{S}|D) \geq 1 - \delta, \tag{S14}$$

where $\mathcal{B}(\theta \in \mathcal{S}|D)$ denotes a measure of the agent's belief that $\theta \in \mathcal{S}$ given data D.

This approach has two problems. The first is that the feasible set $\{\theta \in \Theta : \mathcal{B}(\theta \in \mathcal{S}|D) \geq 1-\delta\}$ is a function of the data, which is a random variable, and so the feasible set is a random variable. A problem formulation in which the feasible set is a random variable suggests that the formulation does not adequately capture the algorithm designer's goals, which are not likely to be random. The second problem with this approach is more serious. This problem formulation could result in unsafe solutions with high probability, even when there is only a single constraint. If we view $\theta^\star$ as a function of the training data D, the probability that $\theta^\star(D)$ is not in $\mathcal{S}$ could be large. It is straightforward to show that (as a consequence of Boole's inequality) if using either definition of $\mathcal{B}$ proposed above or other definitions of $\mathcal{B}$ based on concentration inequalities, the probability that $\theta^\star(D) \notin \mathcal{S}$ can be as large as $\min\{1, \delta|\Theta|\}$, which will be large for problems in which $|\Theta|$ is large. To avoid this, the constraint that one introduces should not require the feasible set to include only solutions believed to be safe; it should instead directly require that the *solution output by the algorithm* be safe with high probability. This is at the heart of the SOP formulation, which we introduce next.

3.3 Seldonian Optimization Problem (SOP)

The first step of the SOP framework is to mathematically define the goal of the researcher creating the machine learning algorithm. This results in a problem formulation that describes a search over algorithms, rather than solutions, with constraints over the set of algorithms, not over the set of solutions. This means that the constraints, now over algorithms, constrain the probability that the algorithm will return an unsafe solution. For simplicity, here we focus on the *batch* setting, where an algorithm has access to all of the available data from the start, as opposed to the *online* setting in which data incrementally arrives over time [88]. We need to define a few terms to precisely state the SOP framework.

A machine learning algorithm a is a function that takes data as input and produces as output a solution. Let $\mathcal{D}$ be the set of all possible data sets that could be given as input to the algorithm, and let $D \in \mathcal{D}$ be a random variable that represents all of the data given to the algorithm (i.e., the training data). Let $\mathcal{A}$ be the set of all possible machine learning algorithms, each of which is a function $a : \mathcal{D} \to \Theta$. This definition of machine learning algorithms allows for probabilistic algorithms since the data set can be defined to contain any random numbers required by the algorithm.

Whereas in the standard ML approach the objective function is a function of possible problem solutions, in an SOP the objective function is a function of algorithms. That is,

$f : \mathcal{A} \to \mathbb{R}$. An SOP allows for $n \in \mathbb{N}_{\geq 0}$ *behavioral constraints*, which are constraints on the set of algorithms. Each constraint, indexed using $i \in \{1, \ldots, n\}$, has two components: a *constraint objective* $g_i : \Theta \to \mathbb{R}$, and a *constraint confidence level* $\delta_i \in [0, 1]$. The designer of the algorithm must ensure that the algorithm he or she creates satisfies the inequalities $\Pr(g_i(a(D)) \leq 0) \geq 1 - \delta_i$, for all $i \in \{1, \ldots, n\}$. That is, if the user defines g_i such that $g_i(\theta)$ being greater than zero means that undesirable behavior has occurred, then an algorithm that satisfies the behavioral constraints will ensure that with high probability (where the user can specify the necessary probability) it will not return a solution that causes undesirable behavior.

In summary, an SOP can be written as:

$$\arg\max_{a \in \mathcal{A}} f(a) \tag{S15}$$

$$\text{s.t. } \forall i \in \{1, \ldots, n\}, \Pr(g_i(a(D)) \leq 0) \geq 1 - \delta_i. \tag{S16}$$

Here the designer of the algorithm selects the objective function f, the set $\mathcal{A}$ of algorithms considered, what the data D will include, and a class of allowable constraint objective functions g_i. (For example, g_i could be the discrimination statistic in Eq. S4, i.e., the expected difference in predictions for men and women. Here the expectation is taken with respect to the distribution of men and women from which the input data D is sampled. Typically a constraint function g_i will be a function of terms including expected values with respect to a particular distribution over data, such as the distribution of D as in Eq. S4.) Once a machine learning algorithm has been designed using this framework, the user of the algorithm can apply it by selecting specific definitions for the g_i from within the class chosen by the researcher who designed the algorithm, and by selecting the desired confidence levels δ_i. In some cases, the user might also have the additional freedom to select other terms, such as the feasible set Θ, or the objective function f.

A common point of confusion is about which terms in Eq. S15 are random. Our guarantees are similar in spirit to *probably approximately correct* (PAC) style guarantees. PAC algorithms provide high probability generalization guarantees on an algorithm's solution, where the high probability is with respect to the randomness in the data set to which the algorithm is applied. Similarly, in our case the data set D is a random variable (the source of randomness in the behavioral constraints), and so $a(D)$ is therefore also a random variable (the model returned when training using data D), as is $g_i(a(D))$ (the amount of undesirable behavior produced by the model that would result from training on data D). The meaning of the behavioral constraints is therefore that, if one where to sample a large number of complete training data sets D from the distribution that produces the training data, and one were to apply algorithm a to obtain a solution (learn a model) for each of these training data sets, one would expect at most roughly $100\delta_i\%$ of the resulting solutions (models) would produce undesirable behavior.

Notice also that whether a solution produces undesirable behavior is a deterministic property of each solution (each model) θ. That is, either $g_i(\theta) \leq 0$ or not. However, one might define $g_i(\theta)$ to be less than or equal to zero if and only if applying θ results in the probability of some undesirable outcome being at most some threshold. For example, if θ corresponds to the weights in a neural network used to classify whether a tumor is benign or malignant, one might define $g_i(\theta) \leq 0$ if and only if using weights θ results in the probability

of a false negative (saying a tumor is benign when it is malignant) being at most 0.1%. The resulting behavioral constraint ensures that the probability of sampling training data that results in a learned model, which causes the probability of false positives to be below 0.1%, is at least $1 - \delta_i$. This guarantee is enabled not by changing the distribution of training data, but by careful computation within the algorithm a.

Notice also that, when defining the constraints for an SOP, the term $\forall i \in \{1, \ldots, n\}$ is outside of the $\Pr(\cdot)$ term. This means that each behavioral constraint must hold independently with its associated probability $1 - \delta_i$. Joint constraints, for example, two different constraints g' and g'' that must hold simultaneously with some probability, can be encoded within a single constraint in the SOP.

Consider how a linear regression algorithm that is a solution to an SOP could be applied to our previous illustrative example. The researcher designing the linear regression algorithm would select f to be an objective function like the MSE, Θ to be the set of all possible linear (or affine) functions, and D to be the training set as described previously. To apply the resulting algorithm, the user creating an agent need only specify g_i and δ_i. For our illustrative example, the user might choose to use a single behavioral constraint $g_1(a(D)) \coloneqq |d(a(D))| - \epsilon$ to guarantee that, with probability at least $1 - \delta_1$, the absolute value of the discrimination statistic will be at most ϵ, where ϵ and δ_1 are chosen by the user.

It is important to observe that to apply this algorithm, the user need not know whether any particular estimator $\theta \in \Theta$ causes $g_1(\theta) \leq 0$. It is left to the algorithm to reason about this. The user need not know the value of $g_i(\theta)$ for any particular θ. Instead, the user only needs to be able to specify g_i. As illustrated by our later reinforcement learning example, the user only needs the ability to recognize undesirable behavior, not which solutions cause undesirable behavior. The job of analyzing the available data to understand a given problem well enough to satisfy the constraint is placed entirely on the machine learning algorithm. Furthermore, notice that although an SOP is more complicated than the problem formulation used in the standard ML approach, and the job of the researcher using our framework to design a machine learning algorithm will likely be more difficult than with the standard ML approach, the job of the user who wishes to constrain the behavior of an agent is dramatically simplified.

3.4 Seldonian and Quasi-Seldonian Algorithms

An SOP, as defined in Eq. S15, prescribes a principled way to specify constraints on an algorithm. However, it has shortcomings that we discuss here and in the following sections. First, finding an *optimal algorithmic solution*, which is an algorithm that maximizes f subject to the behavioral constraints, is generally intractable. However, the researcher designing an algorithm should at least ensure that the algorithms that he or she proposes satisfy the behavioral constraints, setting aside the goal of also maximizing f. We call an algorithm that satisfies all the behavioral constraints a *Seldonian algorithm*.

Although the SOP definition captures what we want in an algorithm, it may not be feasible to find a Seldonian algorithm, let alone an optimal algorithmic solution. Our ability to ensure that an algorithm satisfies a behavioral constraint typically requires the use of *concentration inequalities* [89] like Hoeffding's inequality [90]. Because the confidence bounds produced by these inequalities hold for *any* distribution (given minor verifiable assumptions),

they tend to be overly conservative. This means that all Seldonian algorithms for some problems might require an impractical amount of data before the confidence levels required by the behavioral constraints can be satisfied.

To remedy this, we propose *quasi-Seldonian algorithms*, which are algorithms that satisfy the behavioral constraints of an SOP using approximate concentration bounds. For example, rather than using Hoeffding's inequality, one might use Student's t-test or a bootstrap confidence bound [91]. These methods produce bounds that are often much tighter than those of Hoeffding's inequality, which means that quasi-Seldonian algorithms can be created for problems for which there is insufficient data to find viable Seldonian algorithms. A shortcoming of these approximate concentration bounds is that they rely on assumptions that may not hold for the problem at hand, which means that the resulting confidence bounds only approximately hold. Despite their shortcomings, the use of approximate concentration bounds such as Student's t-test and bootstrap confidence bounds remains commonplace in the sciences, including high-risk medical research [92, 93].

One limitation of quasi-Seldonian algorithms is that the user of the algorithm might not know what false assumptions the algorithm relies upon. A researcher could propose an algorithm with egregious false assumptions that would make it irresponsible to use the algorithm for many applications. To remedy this, researchers should clearly state the assumptions upon which quasi-Seldonian algorithms rely, and should strive to ensure that for most problems the quasi-Seldonian algorithm ensures that $\forall i \in \{1, \ldots, n\}$, $\Pr(g_i(a(D)) \leq 0) \gtrsim 1 - \delta_i$. Furthermore, one should not convert a quasi-Seldonian algorithm into a Seldonian algorithm by placing assumptions on the problems it can be applied to, when these assumptions may not hold for the problems to which it will likely be applied. For example, one should not label a quasi-Seldonian algorithm that relies on Student's t-test as a Seldonian algorithm with the condition that it only be applied to problems where the sum of random variables, the means of which will be bounded using Student's t-test, are normally distributed, when for real problems this sum will only be approximately normally distributed.

Even using approximate confidence bounds, there may not be sufficient data for any algorithm to ensure that the behavioral constraints are met. Rather than require an algorithm to produce a solution even in such cases, we propose allowing the algorithm to return No Solution Found (NSF). The algorithm should return NSF when it is unable to find a satisfactory solution given the available data.

To add NSF to the problem formulation, one need only define Θ such that NSF $\in \Theta$ and g_i such that $g_i(\text{NSF}) \leq 0$ for all i. This modification is sufficient to ensure that Seldonian algorithms exist, but it is most applicable for problems for which returning NSF does not itself cause unsatisfactory agent behavior. This is usually the case for applications in which an existing mechanism (solution) θ_0 is in place for making decisions, and the machine learning algorithm is used to improve upon this mechanism.[2] For these applications, if a Seldonian algorithm returns NSF, we can revert to using the baseline solution. Still, for some applications a decision other than NSF must be made, in which cases the Seldonian problem formulation may not be viable.

Of course, if NSF is included in the SOP, then the algorithm that always returns NSF is

[2]Petrik et al. [94] discuss a similar setting, wherein improvements must be ensured, but the algorithm is free to fall back to an existing prior solution.

trivially Seldonian. If the SOP is properly designed, the objective function f should assign low utility to this trivial algorithm, and so the researcher designing the algorithm should attempt to find a better algorithm: a Seldonian algorithm that returns NSF less frequently.

4 Example: A Seldonian Regression Algorithm and its Application

Here we develop algorithms using the SOP framework to show its viability. We begin by considering the problem of designing a Seldonian linear regression algorithm. Recall that designing a Seldonian linear regression algorithm will be more difficult than designing a linear regression algorithm using the standard ML approach to designing machine learning algorithms, but that it should be easier for the user to constrain an agent's behavior when using the Seldonian algorithm.

We assume that there is a real-valued random variable Y that we would like to predict from a real-vector-valued random variable X. That is, each value $x \in \mathbb{R}^l$ that X can take is a vector of $l \in \mathbb{N}_{>0}$ real-valued features that we will use to estimate the value of Y. We restrict the class of estimators that we consider to linear functions, so $\Theta = \mathbb{R}^l$ and each $\theta \in \Theta$ defines a linear combination of the features $x \in \mathbb{R}^l$, as $\hat{y}(x, \theta) = \theta^\intercal x$. To select the weights θ that cause $\hat{y}(X, \theta)$ to be a good estimate of Y, we are given $m \in \mathbb{N}_{>0}$ realizations of (X, Y). Each realization also comes with a sample of another random variable $T \in \{0, 1\}$ that encodes the *type* (e.g., gender) associated with the point (X, Y). Therefore, the training data is a random variable $D = \{(X_i, Y_i, T_i)\}_{i=1}^m$, where some joint distribution over (X, Y, T) exists but is not known. Notice that although X_i, Y_i, and T_i are all dependent random variables, the tuples (X_i, Y_i, T_i) and (X_j, Y_j, T_j) are independent and identically distributed for all $i \neq j$.

Given this setup, we can write an SOP that defines our goals when designing the regression algorithm:

$$\arg\max_{a \in \mathcal{A}} -\mathbf{E}\left[(\hat{y}(X, a(D)) - Y)^2\right] \tag{S17}$$

$$s.t.\ \forall i \in \{1, \ldots, n\}\ \Pr(g_i(a(D)) \leq 0) \geq 1 - \delta_i. \tag{S18}$$

Importantly, we should allow the user of the algorithm to specify the functions g_i without knowing their values, $g_i(\theta)$, for any particular solution $\theta \in \Theta$. Furthermore, the language provided to the user for specifying the g_i should be as simple as possible, while encompassing a broad spectrum of possible functions. If our goal is to produce a regression algorithm that can be applied to many different regression problems, then the resulting SOP could be defined for a distribution over regression problems—a distribution over joint distributions for X, Y, and T.

Below we present three algorithms: one Seldonian and two quasi-Seldonian. These are not likely to produce optimal algorithmic solutions, but all the solutions they produce satisfy the behavioral constraints while "trying" to maximize the objective function. Although these algorithms can be extended to allow for a broad class of definitions of g_i, we initially restrict our discussion to a single behavioral constraint given by the following constraint objective:

$$g(\theta) = \left|\mathbf{E}\Big[\hat{y}(X, \theta) - Y \Big| T = 0\Big] - \mathbf{E}\Big[\hat{y}(X, \theta) - Y \Big| T = 1\Big]\right| - \epsilon. \tag{S19}$$

Later we present a more general quasi-Seldonian algorithm and derive a quasi-Seldonian reinforcement learning algorithm that showcase how an algorithm can both allow the user to

specify g_i to encode his or her own definition of undesirable behavior and allow for multiple behavioral constraints.

4.1 Non-Discriminatory Linear Regression

First, consider the Seldonian algorithm *non-discriminatory linear regression* (NDLR), presented in Fig. S9, which relies on the subroutines presented in Figs. S5–S8. NDLR has three main steps. First, the data set D is partitioned into two sets, D_1 and D_2, for reasons that we discuss later. Second, D_1 is used to select a single solution, called the *candidate solution* θ_c, that the algorithm considers returning. Third, the algorithm runs a safety test using D_2 to determine whether returning the candidate solution would violate a behavioral constraint. More precisely, the algorithm computes a high confidence upper bound on the absolute value of the discrimination statistic of the candidate solution: a high confidence upper bound on $|d(\theta_c)|$. If this upper bound is less than ϵ, then the candidate solution is returned, and if it is not, then the algorithm returns No Solution Found. Next we discuss in more detail the two main steps of the algorithm, the safety test and the selection of a candidate solution.

The safety test (lines 4–6 of Fig. S9) is the component of the algorithm that ensures that it is Seldonian. Regardless of which candidate solution θ_c is chosen, this step ensures that the behavioral constraint is satisfied. It ensures that with high probability an estimator with absolute discrimination statistic larger than ϵ is not returned. The key component of this step is a method for producing high confidence upper bounds on the absolute discrimination statistic given the data set D_2.

To compute these upper bounds, we use `HoeffdingDiscrimUpperBound` (Fig. S8), which relies on one form of Hoeffding's inequality [90], which is provided in Fig. S5, `HoeffdingUpperBound`. Hoeffding's inequality is not directly applicable due to the absolute value in the absolute discrimination statistic. To remedy this, we replace the single behavioral constraint that the absolute discrimination statistic is less than ϵ with two behavioral constraints that ensure that $d(a(D)) \geq -\epsilon$ and $d(a(D)) \leq \epsilon$ with high probability. Because these two constraints must hold simultaneously with probability $1 - \delta$, we require each to hold with probability $1 - \delta/2$.

1 **return** $\frac{1}{m}\left(\sum_{i=1}^{m} Z_i\right) + b\sqrt{\frac{\ln(1/\delta)}{2m}}$;

Fig. S5: `HoeffdingUpperBound`(Z, b, δ): Apply Hoeffding's inequality to upper bound the mean of a random variable. $Z = \{Z_1, \ldots, Z_m\}$ is the vector of samples of the random variable, $b \in \mathbb{R}_{\geq 0}$ is the range of the random variable, and $\delta \in (0, 1)$ is the confidence level of the upper bound.

return $\frac{1}{m}\left(\sum_{i=1}^{m} Z_i\right) + b\sqrt{\frac{\ln(1/\delta)}{2k}}$;

Fig. S6: `PredictHoeffding`(Z, b, δ, k): Predict what would be returned by `HoeffdingUpperBound`(Z', b, δ) if given a new array of samples Z' with k elements. We use a conservative prediction (an over-prediction) because under-predictions can cause NDLR to frequently return No Solution Found when it could safely return a solution.

Input: 1) Solution θ, **2)** data set $D = \{(X_i, Y_i, T_i)\}_{i=1}^{m}$, **3)** confidence level $\delta \in (0, 1)$, **4)** maximum level of discrimination $\epsilon \in \mathbb{R}_{>0}$, **5)** an upper bound b on the range of possible prediction errors, **6)** a number of samples k.

$m_1 \leftarrow \sum_{i=1}^{m} T_i, \qquad m_0 \leftarrow m - m_1$;

Let (X_i^0, Y_i^0) be the i^{th} data point of type zero, and (X_i^1, Y_i^1) be the i^{th} data point of type one;

Create array Z of length $\min\{m_0, m_1\}$, with values $Z_i = (\theta^\intercal X_i^0 - Y_i^0) - (\theta^\intercal X_i^1 - Y_i^1)$;

$\texttt{ub} = \max\{\texttt{PredictHoeffding}(Z, b, \delta/2, k), \texttt{PredictHoeffding}(-Z, b, \delta/2, k)\}$;

if $ub \leq \epsilon$ **then**

  return $\frac{1}{m}\sum_{i=1}^{m}(\theta^\intercal X_i - Y_i)^2$;

return $b^2 + \texttt{ub} - \epsilon$;

Fig. S7: `HoeffdingCandidateObjective`$(\theta, D, \delta, \epsilon, b, k)$: The objective function maximized by the candidate solution when using Hoeffding's inequality. Here k is the number of data points that will be used in the subsequent safety test in NDLR—the cardinality of D_2.

Input: 1) Solution θ, **2)** data set $D = \{(X_i, Y_i, T_i)\}_{i=1}^{m}$, **3)** confidence level $\delta \in (0, 1)$, **4)** maximum level of discrimination $\epsilon \in \mathbb{R}_{>0}$, **5)** an upper bound b on the range of possible prediction errors.

$m_1 \leftarrow \sum_{i=1}^{m} T_i, \qquad m_0 \leftarrow m - m_1$;

Let (X_i^0, Y_i^0) be the i^{th} data point of type zero, and (X_i^1, Y_i^1) be the i^{th} data point of type one;

Create array Z of length $\min\{m_0, m_1\}$, with values $Z_i = (\theta^\intercal X_i^0 - Y_i^0) - (\theta^\intercal X_i^1 - Y_i^1)$;

return $\max\{\texttt{HoeffdingUpperBound}(Z, b, \delta/2), \texttt{HoeffdingUpperBound}(-Z, b, \delta/2)\}$;

Fig. S8: `HoeffdingDiscrimUpperBound`$(\theta, D, \delta, \epsilon, b)$: Compute a high-probability upper bound on the discrimination statistic using Hoeffding's inequality.

1 **Input:** **1)** Data set $D = \{(X_i, Y_i, T_i)\}_{i=1}^{m}$, **2)** confidence level $\delta \in (0, 1)$, **3)** maximum level of discrimination $\epsilon \in \mathbb{R}_{>0}$, **4)** an upper bound b on the magnitude of a prediction error.
2 Partition D into D_1 (20% of data) and D_2 (80% of data);
3 $\theta_c \in \arg\min_{\theta \in \Theta} \texttt{HoeffdingCandidateObjective}(\theta, D_1, \delta, \epsilon, b, |D_2|)$;
4 **if** *HoeffdingDiscrimUpperBound*$(\theta_c, D_2, \delta, \epsilon, b) \leq \epsilon$ **then**
5 **return** θ_c;
6 **return** No Solution Found;

Fig. S9: Non-Discriminatory Linear Regression (NDLR, "endler").

Next consider the selection of the candidate solution, θ_c, in more detail (line 3 of Fig. S9). Poor choices of θ_c will result in the algorithm returning No Solution Found. Good choices of θ_c will be solutions that will be returned by the third step of NDLR, which are solutions whose discrimination statistics will be successfully bounded below ϵ when using D_2. Furthermore, good choices of θ_c should also minimize the MSE of their predictions. The goal is not just to satisfy the behavioral constraint, but also to optimize the objective, in this case to minimize the MSE. Consequently, NDLR (Fig. S9) selects the candidate solution that it predicts (on the basis of D_1) will have the lowest MSE subject to the constraint that it is predicted that the candidate solution will be returned. To accomplish this optimization, NDLR relies on `HoeffdingCandidateObjective` (Fig. S7), which uses a boundary function to constrain the search to solutions that are predicted to be returned in the third step.

Notice that the selection of the candidate solution and the safety test use different and statistically independent data sets, D_1 and D_2. This is necessary to ensure statistical independence of the random variables used by Hoeffding's inequality when computing high confidence bounds on the discrimination statistic. Partitioning the data ensures that the candidate solution makes no use of the data that will be used to test its safety. Access to this data could bias the results of the safety test.

4.2 Quasi-Non-Discriminatory Linear Regression

Next consider the quasi-Seldonian regression algorithms *quasi-non-discriminatory linear regression* (QNDLR) and QNDLR(λ) (Fig. S14), which rely on the subroutines presented in Figs. S10–S13. QNDLR and QNDLR(λ) are modifications of NDLR that use Student's t-test, `TTestUpperBound` in Fig. S10, rather than Hoeffding's inequality when computing high confidence upper bounds.

Both NDLR and QNDLR attempt to minimize the MSE while ensuring that the discrimination statistic is at most ϵ with high probability. Given large amounts of data (e.g., as $m \to \infty$), they tend to converge to solutions with discrimination statistics slightly less than ϵ. However, the actual goal is not to produce solutions with discrimination statistics close to ϵ; the goal is to avoid discrimination as much as possible. That is, this problem is really a multiobjective problem: the ideal solution should simultaneously minimize MSE and the discrimination statistic.

$\bar{Z} \leftarrow \frac{1}{m}\sum_{i=1}^{m} Z_i, \qquad \sigma \leftarrow \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(Z_i - \bar{Z})^2};$

return $\bar{Z} + \frac{\sigma}{\sqrt{m}} t_{1-\delta,m-1}$;

Fig. S10: `TTestUpperBound`(Z, δ): Apply Student's t-test to a vector of samples of a random variable. $Z = \{Z_1, \ldots, Z_m\}$ is the vector of samples and $\delta \in (0, 1)$ is the confidence level.

$\bar{Z} \leftarrow \frac{1}{m}\sum_{i=1}^{m} Z_i, \qquad \sigma \leftarrow \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(Z_i - \bar{Z})^2};$

return $\bar{Z} + 2\frac{\sigma}{\sqrt{k}} t_{1-\delta,k-1}$;

Fig. S11: `PredictTTest`(Z, δ, k): Predict what `TTestUpperBound`(Z, δ) would return if given a new array of samples Z with k elements, and err on the side of over-estimating. We use a conservative prediction (an over-prediction) because under-predictions can cause NDLR to frequently return No Solution Found.

Input: 1) Solution θ, **2)** data set $D = \{(X_i, Y_i, T_i)\}_{i=1}^{m}$, **3)** confidence level $\delta \in (0, 1)$, **4)** maximum level of discrimination $\epsilon \in \mathbb{R}_{>0}$, **5)** a number of samples k, **6)** a constant $\lambda \in \mathbb{R}_{\geq 0}$ that balances the trade-off between MSE and discrimination.

$m_1 \leftarrow \sum_{i=1}^{m} T_i, \qquad m_0 \leftarrow m - m_1;$

Let (X_i^0, Y_i^0) be the i^{th} data point of type zero, and (X_i^1, Y_i^1) be the i^{th} data point of type one;

Create array Z of length $\min\{m_0, m_1\}$, with values $Z_i = (\theta^\intercal X_i^0 - Y_i^0) - (\theta^\intercal X_i^1 - Y_i^1)$;

$\texttt{ub} = \max\{\texttt{PredictTTest}(Z, b, \delta/2, k), \texttt{PredictTTest}(-Z, b, \delta/2, k)\}$;

if $ub \leq \epsilon$ **then**

  return $\frac{1}{m}\sum_{i=1}^{m}(\theta^\intercal X_i - Y_i)^2 + \lambda\frac{1}{|Z|}\sum_{i=1}^{|Z|}|Z_i|$;

return $b^2 + \texttt{ub} + (\lambda - 1)\epsilon$;

Fig. S12: `TTestCandidateObjective`$(\theta, D, \delta, \epsilon, k, \lambda)$: The objective function maximized by the candidate solution.

Input: 1) Solution θ, **2)** data set $D = \{(X_i, Y_i, T_i)\}_{i=1}^{m}$, **3)** confidence level $\delta \in (0, 1)$, **4)** maximum level of discrimination $\epsilon \in \mathbb{R}_{>0}$.

$m_1 \leftarrow \sum_{i=1}^{m} T_i, \qquad m_0 \leftarrow m - m_1;$

Let (X_i^0, Y_i^0) be the i^{th} data point of type zero, and (X_i^1, Y_i^1) be the i^{th} data point of type one;

Create array Z of length $\min\{m_0, m_1\}$, with values $Z_i = (\theta^\intercal X_i^0 - Y_i^0) - (\theta^\intercal X_i^1 - Y_i^1)$;

return $\max\{\texttt{TTestUpperBound}(Z, \delta/2), \texttt{TTestUpperBound}(-Z, \delta/2)\}$;

Fig. S13: `TTestDiscrimUpperBound`$(\theta, D, \delta, \epsilon)$: Compute a high-probability upper bound on the discrimination statistic using Student's t-test.

1 **Assumptions:** This quasi-Seldonian algorithm assumes that the sample discrimination statistic computed using all of the data is normally distributed, and also uses the minimum MSE estimator of variance within Student's t-test rather than the unbiased estimator.
2 **Input: 1)** Data set $D = \{(X_i, Y_i, T_i)\}_{i=1}^m$, **2)** confidence level $\delta \in (0, 1)$, **3)** maximum level of discrimination $\epsilon \in \mathbb{R}_{>0}$, **4)** a hyperparameter $\lambda \in \mathbb{R}_{\geq 0}$.
3 Partition D into D_1 (20% of data) and D_2 (80% of data);
4 $\theta_c \in \arg\min_{\theta \in \Theta}$ `TTestCandidateObjective`$(\theta, D_1, \delta, \epsilon, |D_2|, \lambda)$;
5 **if** *`TTestDiscrimUpperBound`*$(\theta_c, D_2, \delta, \epsilon) \leq \epsilon$ **then**
6 **return** θ_c;
7 **return** No Solution Found;

Fig. S14: Quasi-Non-Discriminatory Linear Regression (QNDLR) if $\lambda = 0$ and QNDLR(λ) if $\lambda > 0$.

This is an example of a problem for which multiobjective methods can be combined with the SOP framework. QNDLR(λ) is a modification of QNDLR to allow for a multiobjective approach while maintaining the high-probability guarantees of a quasi-Seldonian algorithm. Specifically, QLDNR(λ) is a variant of QNDLR that includes a soft constraint parameterized by λ in the objective function as in Eq. S7. QNDLR(λ) is therefore a solution to the Seldonian optimization problem in Eq. S17, modified so that the MSE objective includes a penalty proportional to the sample discrimination statistic. QNDLR(λ) is a quasi-Seldonian algorithm that ensures with high probability that the discrimination statistic will be at most ϵ, and subject to this constraint, it simultaneously optimizes MSE and the discrimination statistic.

Recall that selecting λ is a difficult process that often requires additional data analysis, and that values of λ that are too small will result in a standard soft-constrained method producing a solution with discrimination statistic larger than ϵ. Crucially, because QNDLR(λ) is a quasi-Seldonian algorithm, even if the user selects a λ that would result in discrimination statistics larger than ϵ if using a standard soft-constrained method, QNDLR(λ) will, with high probability, not return a solution with discrimination statistic larger than ϵ. This effectively eliminates the risk associated with selecting a value for λ that would make standard soft-constrained methods unsafe,

Lastly, it is clear that NDLR, QNDLR, and QNDLR(λ) have significantly higher computational complexity than ordinary least squares linear regression and many multiobjective methods. The primary computational bottleneck in our algorithms is the search for the solution θ_c that optimizes the candidate objective function (lines 3 and 4 in Fig. S9 and Fig. S14, respectively). This optimization includes hard constraints, which we transformed into soft constraints using boundary functions. In general, solving optimization problems with hard constraints is more challenging than solving problems with soft constraints (a computational benefit of soft-constrained and multiobjective methods). Different choices of methods for performing the search for θ_c result in different computational complexities. In our experiments we performed the search using gradient descent with error-based termination conditions. Although the higher computational complexity of the (quasi-)Seldonian algorithms that we present was not an issue in our experiments, for big-data applications it

could be a concern. Thus it remains an important open question whether the search for a candidate solution can be further optimized computationally.

4.3 NDLR and QNDLR Discussion

NDLR, QNDLR, and QNDLR(λ) are instances of a more general algorithm that allows the user to **1)** select any objective function (e.g., MSE alone as used by NDLR and QNDLR, or MSE with a penalty proportional to the discrimination statistic as used by QNDLR(λ)), and **2)** bound any statistic for which the user can provide data-based unbiased estimates. That is, the user provides **1)** a function $\hat{f}$, such that $\hat{f}(\theta, D) \in \mathbb{R}$ is an estimate of the utility of the solution θ, computed using data D and **2)** a function $\hat{g}$, such that $\hat{g}(\theta, D) \in \mathbb{R}^{|D|}$ is a vector of i.i.d. unbiased estimates of the desired behavioral constraint function $g(\theta)$ (thus, $g(\theta) \coloneqq \mathbf{E}[\hat{g}_1(\theta, D)]$). This more general quasi-Seldonian algorithm, presented in Fig. S15, is an approximate solution to the Seldonian optimization problem:

$$\arg\max_{a \in \mathcal{A}} \mathbf{E}\left[\hat{f}(a(D), D')\right] \tag{S20}$$

$$\text{s.t.} \quad \Pr(\mathbf{E}_{D'}[\hat{g}_1(a(D), D')] \leq 0) \geq 1 - \delta, \tag{S21}$$

where the D and D' data sets are independent and identically distributed random variables and $\mathbf{E}_{D'}$ denotes that the expected value is taken only over D' (not D).

Intuitively, the quasi-Seldonian algorithm presented in Fig. S15 is similar to QNDLR—it partitions the data set, uses one partition to compute a candidate solution, and uses Student's t-test with the second partition to test whether the candidate solution can be returned. In fact, QNDLR is a special case of this algorithm, where $\hat{f}$ is the negative sample MSE and $\hat{g}$ is the vector of sample discrimination statistics denoted by Z in Fig. S13. It is straightforward to extend the algorithm in Fig. S15 to allow for multiple behavioral constraints and to allow for functions $\hat{g}$ that produce variable numbers of outputs.

Notice that this more general algorithm could be used to bound other notions of discrimination. For example, one could define $\hat{g}$ to be the difference in predictions (rather than prediction errors) to require the mean predictions to be similar for people of each type. In this sense, users are free to select the definitions of discrimination that they each desire. Moreover, users need not know the true values of the statistics that they wish to ensure are bounded. As an example of this, notice that specifying the $\hat{g}$ function used by QNDLR does not require knowledge of the true discrimination statistic of any solutions.

Lastly, consider what would happen if the user asked for the impossible: what if the user desired the difference in mean predictions for people of each type, as well as the difference in mean prediction errors for people of each type, to both be small? For our illustrative example it is straightforward to show that this is not possible. As a result, any Seldonian algorithm, including the one in Fig. S15, should return No Solution Found with high probability—that is, a (quasi-)Seldonian algorithm can effectively say "I cannot do that."

We now discuss a different point: given small amounts of data, both NDLR and QNDLR may often return No Solution Found. This raises the question: what should one do if faced with a problem where there is not sufficient data to guarantee (even using approximate concentration bounds) that the resulting solution will not produce discriminatory behavior? One tempting solution is to argue that, in this case, one should ignore the behavioral

Input : • Feasible set Θ, data set D, and probability $1 - \delta$.
• Function $\hat{f}$ such that $\hat{f}(\theta, D) \in \mathbb{R}$ is an estimate of the utility of the solution θ, computed using data D.
• Function $\hat{g}$, such that $\hat{g}(\theta, D) \in \mathbb{R}^{|D|}$ is a vector of unbiased estimates of $g(\theta)$.

Output : A solution, $\theta \in \Theta$, or No Solution Found.

Partition D into two data sets, D_1 and D_2;

$\theta_c = \arg\max_{\theta \in \Theta} \hat{f}(\theta, D_1)$ s.t. $\mu(\hat{g}(\theta, D_1)) + 2\frac{\sigma(\hat{g}(\theta, D_1))}{\sqrt{|D_2|}} t_{1-\delta,|D_2|-1} \leq 0$;

if $\mu(\hat{g}(\theta_c, D_2)) + \frac{\sigma(\hat{g}(\theta_c, D_2))}{\sqrt{|D_2|}} t_{1-\delta,|D_2|-1} \leq 0$ **then**

  return θ_c;

return No Solution Found;

Fig. S15: An example of a general-purpose quasi-Seldonian algorithm. Here $\mu(v)$ denotes the average of the elements of the vector v and $\sigma(v)$ denotes the sample standard deviation of the elements of the vector v including Bessel's correction.

constraints and simply use ordinary least squares linear regression. However, we contend that insufficient data does not require us use methods that do not provide guarantees about their behavior.

Instead, one might use algorithms that relax the behavioral constraints even more than quasi-Seldonian algorithms. Intuitively, Seldonian algorithms ensure that the probability of undesirable behavior is at most δ, while quasi-Seldonian algorithms ensure that the probability of undesirable behavior will be less then or approximately equal to δ. One might go a step further and define even weaker variants of quasi-Seldonian algorithms with other properties. For example, one might require the algorithm to produce solutions in a way such that a third party could not show negligence—so that someone else could not use the data available to the algorithm to show with high confidence that the algorithm produced a solution whose discrimination statistic has magnitude greater than ϵ. However, here we focus primarily on quasi-Seldonian algorithms. Seldonian algorithms like NDLR can require too much data to be practical, and while weaker variants of quasi-Seldonian algorithms can produce solutions with any amount of data, they do not provide satisfying guarantees about their behavior. Quasi-Seldonian algorithms like QNDLR provide a nice middle-ground between these two extremes—they can produce solutions given practical amounts of data, and they also provide the user with practical insights into the probability that undesirable behavior might occur.

4.4 Related Work on Fairness for Supervised Learning

Our NDLR, QNDLR, and QNDLR(λ) are not the first supervised learning algorithms that have been proposed as a means to preclude discriminatory behavior or ensure fairness. However, to the best of our knowledge, they are the first to provide the user with a practical guarantee about the probability of undesirable behavior (discrimination). In particular, although some existing fairness-aware methods provide upper bounds on the severity of unfair behavior [24, 51], they do not bound the probability that they will return models

which produce unfair behavior in excess of a user-defined tolerance level, which is the guarantee provided by our Seldonian algorithms. As we discuss later, the bounds the existing fairness-aware methods do provide could be used when creating Seldonian algorithms.

Kamiran and Calders [95] provide classification algorithms that aim to ensure that when, e.g., making predictions for people, the probability of a desired prediction is similar regardless a person's type. In the context of linear regression, this would be similar to using a different definition of the discrimination statistic that considers the difference between the mean predictions for people of each type, rather than the difference between the mean prediction *errors*:

$$d(\theta) \coloneqq \mathbf{E}\left[\hat{y}(X,\theta)|T=0\right] - \mathbf{E}\left[\hat{y}(X,\theta)|T=1\right]. \tag{S22}$$

There have been many extensions and adaptations of that approach [96, 97, 98, 50, 99, 100, 101], and there is an extensive history of related game-theoretic efforts [102, 103, 104].

There have also been recent efforts to solve the related problem of determining whether a deployed machine learning algorithm is producing discriminatory behavior (as opposed to designing non-discriminatory algorithms). For example, research has proposed means for determining how sensitive a black-box supervised learning algorithm is to each feature in a vector of features used to represent the input [105, 106], which can be used to determine the impact of features, such as race and gender [107]. Automated testing can similarly be applied to software systems that rely on learned models to measure fairness [108].

Our work allows for a large number of different definitions of undesirable behavior, including many definitions of fairness or discrimination. However, the primary difference between our work and prior related work is *not* our differing definitions of discrimination. We selected the discrimination statistic presented in Eq. S4 as an example, which we used to create NDLR and QNDLR as simple first examples of (quasi-)Seldonian algorithms. Both NDLR and QNDLR are special cases of the more general (quasi-)Seldonian algorithm presented in Fig. S15, which allows its users to specify their own definitions of undesirable behavior (in this case, discrimination). That is, the algorithm in Fig. S15 allows the user to select the definition of discrimination that is most suitable for an application. This is in stark contrast to the aforementioned methods for algorithmic fairness that each only allow the user to mitigate a particular form of discrimination.

To further clarify this point, consider again our illustrative example. For this example, it is straightforward to verify that there does not exist a single estimator that bounds both the difference in mean predictions (see Eq. S22) and the difference in mean prediction errors (see Eq. S4) to both be small. However, the user of the algorithm in Fig. S15 *could* define behavioral constraints to require that both of these statistics be bounded below a small constant with high probability. All (quasi-)Seldonian algorithms, including the one in Fig. S15, should return No Solution Found with high probability in this case. This is effectively the algorithm's way of stating "I cannot do what was requested." Existing methods for algorithmic fairness do not have this capability because they do not give their users the freedom to select their own desired definitions of undesirable behavior (discrimination) from within a sufficiently broad class.

4.4.1 Fairness for Classification

The algorithmic fairness community has been rapidly growing and evolving. Part of this growth has been an increased focus on *classification* algorithms. While the linear regression setting provides an easily accessible example of how undesirable behavior can occur and be mitigated, fairness can also be measured by considering the impact of decisions that are influenced by the predictions made by classifiers. The classification setting is the same as the previously defined regression setting, with random variables X, Y, and T, but where Y is restricted to being in a (typically small) discrete set. Here, we will focus on *binary classification*, wherein $Y \in \{-1, +1\}$. We refer to X as the *feature vector* and we refer to Y as the *label*.

There have emerged numerous definitions of algorithmic fairness [109]. While each of these definitions is appropriate in a given context, many are impossible to satisfy simultaneously [110, 111]. This section reviews recent work on defining fairness, and on developing fair classification algorithms. Later, to show the generality of our approach, we apply our Seldonian regression algorithms (modified to perform classification rather than regression) to enforce several different definitions of fairness, and we present empirical comparisons to some of these related fair classification algorithms.

We explain fairness definitions in terms of a classification model that classifies feature vectors as either a member of the positive ($+1$) or the negative (-1) class. For example, a model that predicts recidivism [3] may classify each individual as either likely to be a repeat offender ($+1$) or unlikely to be a repeat offender (-1). We consider the model's fairness with respect to the protected attribute, $T \in \{0, 1\}$, as before. Although our methods extend to definitions of fairness across non-binary protected attributes, this extension is beyond the scope of this paper. We refer to the set of all feature vectors with the same value for the protected attribute as a *group*. We use the notation $\Pr(\hat{y}(X, \theta){=}{+1}|T{=}\tau)$ to mean the fraction of feature vectors in a group that the model classifies as members of the positive class. We now enumerate definitions of fairness (all of which a Seldonian algorithm could enforce).

- **Disparate treatment**, a concept of legal origins, has been interpreted by the machine learning community in a way that does not always align with the legal definition.[3] For a model to satisfy the machine learning community's definition of disparate treatment with respect to a set of attributes, it must have been learned without access to those attributes [25]. Unfortunately, because data attributes are often correlated, e.g., age correlates with savings, race correlates with name, and, in the United States, race correlates with zip code, models trained without access to a set of attributes can still effectively act unfairly with respect to those attributes [113, 114].

- **Disparate impact**, also a legal concept that has been adapted by the machine learning community, captures the notion that a model that does not consider a set of attributes may still act unfairly with respect to those attributes (e.g., because of correlations among attributes). Such practices could appear to be fair on their face, but, nonetheless,

[3] Kim [112] discusses the subtleties that arise when applying the existing legal doctrines of *disparate treatment* and *disparate impact* to decisions made by machine learning agents.

have adverse effects on the involved groups [115, 116, 25]. To measure disparate impact, the US Supreme Court applied the *The 80% Rule* (previously developed by the Technical Advisory Committee on Testing assembled by the State of California Fair Employment Practice Commission in 1971), which states that an employer's hiring rates for protected groups may not differ by more than 80%. For example, if an employer hires 1/2 of its male applicants, then that employer must hire at least $80\% \cdot \frac{1}{2} = \frac{2}{5}$ of its female applicants [115]. Formally, a classifier that is fair with respect to this disparate impact proxy if

$$\min\left(\frac{\Pr(\hat{y}(X,\theta)=+1|T=0)}{\Pr(\hat{y}(X,\theta)=+1|T=1)}, \frac{\Pr(\hat{y}(X,\theta)=+1|T=1)}{\Pr(\hat{y}(X,\theta)=+1|T=0)}\right) \geq \frac{p}{100}, \tag{S23}$$

where typically $p = 80$, per the 80% rule.

- **Delayed impact** is concerned with the fact that making seemingly fair decisions can, in the long term, produce unfair consequences [117]. For example, to make up for a disparity in recidivism predictions by race, a model may, at random, decrease its predictions for one race. While on its face, this may improve the situation for members of that race, if this results in more visibility for repeat offenders of that race, the public's perception may have a more negative effect toward that race, producing delayed negative impact. Measuring delayed impact requires temporal indicator data, of, for example, long-term improvement, stagnation, and decline in variables of interest [117].

- **Demographic parity**, also called **statistical parity** and **group fairness**, requires that the model's predictions are statistically independent of the attribute with respect to which the model is fair [50, 96]. For example, for a model that predicts an individual's recidivism to be fair with respect to race, it should predict the same fraction of the individuals of each race as likely to be repeat offenders. Formally, demographic parity is satisfied if

$$\Pr(\hat{y}(X,\theta)=+1|T=0) = \Pr(\hat{y}(X,\theta)=+1|T=1). \tag{S24}$$

- **Predictive equality** requires that false positive rates are equal among groups [116, 118]. Formally, predictive equality is satisfied if

$$\Pr(\hat{y}(X,\theta)=+1|T=0, Y=-1) = \Pr(\hat{y}(X,\theta)=+1|T=1, Y=-1). \tag{S25}$$

  Note that this definition only considers feature vectors whose true label is -1.

- **Equal opportunity** requires that false negative rates are equal among groups [119, 116]. Formally, equal opportunity is satisfied if,

$$\Pr(\hat{y}(X,\theta)=-1|T=0, Y=+1) = \Pr(\hat{y}(X,\theta)=-1|T=1, Y=+1). \tag{S26}$$

  Note that this definition only considers feature vectors whose true label is $+1$.

- **Equalized odds**, a combination of predictive equality and equal opportunity, requires that both false positive and false negative rates are equal among groups [119]. Consequently, the equalized odds criterion can be viewed as the conjunction of the predictive equality and equal opportunity criteria.

- **Treatment equality** requires that the ratio of the false-positive rate to the false-negative rate is the same for each group [120]. Formally, treatment equality is satisfied if

$$\frac{\Pr(\hat{y}(X,\theta)=-1|T=0,Y=+1)}{\Pr(\hat{y}(X,\theta)=+1|T=0,Y=-1)}=\frac{\Pr(\hat{y}(X,\theta)=-1|T=1,Y=+1)}{\Pr(\hat{y}(X,\theta)=+1|T=1,Y=-1)}. \tag{S27}$$

- **Causal fairness**, also called **counterfactual fairness**, is based on the counterfactual causal relationship between variables. To be causally fair, a classifier must predict the same label for all feature vectors that are the same except for those attributes. In other words, if two contexts differ only in T, and are otherwise identical, this definition requires classifiers to predict the same outcome for both individuals [108, 121]. For example, a recidivism model is causally fair with respect to race only if it predicts identical labels for all pairs of individuals identical in every way except race.

- **Metric fairness** requires that, given a distance metric to compare two feature vectors, the model should predict similar labels for similar feature vectors, on average [50]. Rothblum and Yona [122] extended this definition by introducing **approximate metric fairness**, which incorporates a tolerance parameter, γ, to obtain PAC-style generalization bounds on metric fairness.

- **Representation disparity** limits the error for all subgroups [51]. Formally, the amount of representation disparity is the maximum loss for any particular group, i.e.,

$$\max_{\tau\in\{0,1\}} \mathbf{E}[\ell(X,\theta)|T=\tau], \tag{S28}$$

  where $\ell(X,\theta)$ is the loss associated with the parameter vector, θ. This could be converted into a constraint by requiring representation disparity to be below a threshold.

- **Conditional use accuracy equality** requires that precision (the probability that the model is correct when it predicts the label, +1) is the same for all groups, and that the negative predictive value (the probability that the model is correct when it predicts the label, –1) is the same for all groups [120]. Formally, conditional use accuracy equality is satisfied if

$$\Pr(Y=+1|T=0,\hat{y}(X,\theta)=+1)=\Pr(Y=+1|T=1,\hat{y}(X,\theta)=+1) \tag{S29}$$

  and

$$\Pr(Y=-1|T=0,\hat{y}(X,\theta)=-1)=\Pr(Y=-1|T=1,\hat{y}(X,\theta)=-1). \tag{S30}$$

- **Overall accuracy equality** requires that the accuracy of the classifier (fraction of the feature vectors that the model correctly classifies) is equal for each group [120]. Formally, overall accuracy equality is satisfied if

$$\Pr(Y=\hat{y}(X,\theta)|T=0)=\Pr(Y=\hat{y}(X,\theta)|T=1). \tag{S31}$$

Using the above definitions, researchers have aimed to build classifiers that make fair decisions. Several methods have emerged for enforcing specific definitions of fairness. However, unlike our approach, these methods do not provide probabilistic guarantees that the resulting classifier is acceptably fair when applied to unseen data. For example, logistic-regression-based classification, support vector machines, and hinge loss classifiers can enforce a convex surrogate for disparate impact by introducing constraints related to the covariance between T and the model's predictions $\hat{y}(X, \theta)$ [25]. This approach is guaranteed to be fair with respect to the surrogate definition, but by contrast to our approach, cannot guarantee fairness with respect to the actual definition of disparate impact.

Similarly, for decision trees, constraining the splitting criteria using T can improve demographic parity [123], as can balancing, removing, and repeating some training data, and flipping output labels of feature vectors close to the decision tree's decision boundaries, which introduces noise around those critical boundaries [124]. Balancing the training data and introducing noise by flipping the labels of some feature vectors and repeating some training data can improve demographic parity for naïve Bayes classifiers as well [125]. More generally, training separate classifiers for each group can limit several types of unfair behavior, including violations of demographic parity [126]. Notably, this requires knowledge of group membership for each feature vector, violating the definition of disparate treatment. In addition, this approach incorporates fairness by augmenting the classification loss with a term that measures unfair behavior, and thus does not guarantee that the returned solution will meet the user's fairness requirements. Directly minimizing the violation of demographic parity through the use of a specialized regularizer is also effective for enforcing fairness [98]. By contrast to our approach, these methods do not guarantee fairness of the classifiers they produce.

While many classifiers are tailored to enforce specific definitions of fairness, there has been recent interest in developing more general methods. For example, one such method, Fairlearn, is able to enforce all fairness constraints that can be formulated as linear functions of conditional moments, such as false-positive rates for specific values of T [24]. This method provides probabilistic upper bounds on the severity of unfair behavior—it can estimate how unfair the classifier is. However, it does not use these bounds to constrain the probability that it creates unfair classifiers. Fairlearn, and the bounds that it does provide, could be used as components of a more advanced Seldonian algorithm that rejects solutions that are predicted to exhibit unfair behavior in excess of the user's tolerance.

Finally, the problem of enforcing fairness has been investigated in several other problem settings besides regression and classification. While there are significant differences between these settings and the ones we consider here, there are some shared concepts. For example, fairness has been formulated in the multi-armed bandit setting by associating each bandit arm with a population, and stating that an algorithm is unfair if it preferentially chooses less-qualified (i.e., lower value) arms [101]. While this definition of fairness is distinct to the setting of multi-armed bandits, the use of confidence intervals to provide high-probability bounds on unfair behavior is similar to our approach.

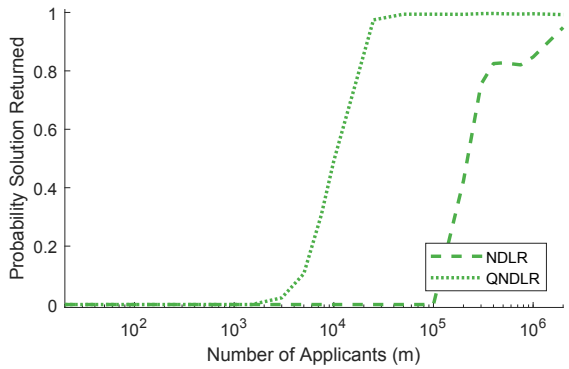
Other recent work has evaluated fairness in the context of repeated interactions with a classification system [51]. Specifically, *disparity amplification* may arise for services that retrain an underlying model iteratively, over time. When such a service exhibits bias against a group, and individuals interact with such a service, members of that group may self-select to discontinue using the service. When the service retrains its model iteratively, a smaller

portion of the training data represents that group, and the bias may increase. While a similar effect could be achieved using our approach by defining fairness based on the predicted outcome of repeated interactions with the classifier, we do not investigate this in the current work. In addition, this approach uses upper bounds on the severity of unfair behavior, which, while distinct from the bounds we provide, might be used to design a related Seldonian algorithm. Finally, other work has introduced definitions of fairness that are tailored to specific applications, such as for use in evaluating recommendation systems [127]. Because these definitions are specialized, we do not list them here. However, our framework could be used to enforce these constraints as well.

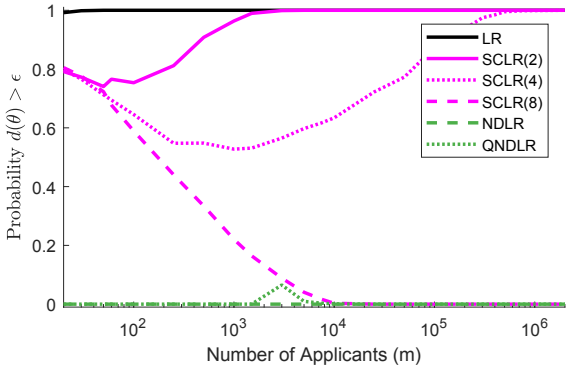
4.5 Application to the Illustrative Example

We now present empirical results from the application of Seldonian and non-Seldonian algorithms. We begin with the application of SCLR, NDLR, and QNDLR to the illustrative example, using $\delta = 0.05$, $\epsilon = 0.1$, and using different amounts, m, of training data. The results are summarized in Fig. S16. Notice that NDLR is extremely conservative—not once did it return a solution with discrimination statistic larger than ϵ. The main limitation of NDLR is that it requires a large amount of data (around $m = 300{,}000$) before it begins to return a solution more often than not. QNDLR is not quite as conservative as NDLR, although it still maintains an error probability less than δ, and requires only around $m = 10{,}000$ training points before it begins to return a solution more often than not. This is an example where QNDLR provides a nice balance of the trade-off between data efficiency (the amount of data needed to find solutions) and how much its probabilistic guarantees can be trusted. Notice also that both NDLR and QNDLR produce solutions that have higher MSE than the ordinary least squares fit. This is because, as discussed previously, minimizing MSE and the absolute discrimination statistic are at odds—the algorithms must balance the trade-off between error and discrimination.

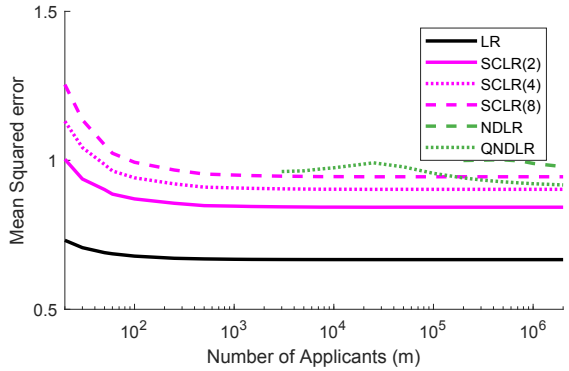
We also provide empirical comparisons to a few other algorithms. First, we include *soft constrained linear regression* (SCLR), which includes soft constraints as described previously using various settings of λ (we write SCLR(λ) to denote SCLR applied with a specific value of λ). Notice that larger values of λ result in smaller magnitude discrimination statistics, and that the magnitude of the discrimination statistic is sensitive to the choice of λ—if it is selected to be too large, the MSE is unnecessarily high, while if it is too small, it will result in too much discrimination. Furthermore, notice that even when using the setting of λ that results in the mean absolute discrimination statistic being approximately ϵ, SCLR always returns a solution, and so given small amounts of data it can often produce solutions that discriminate too much.



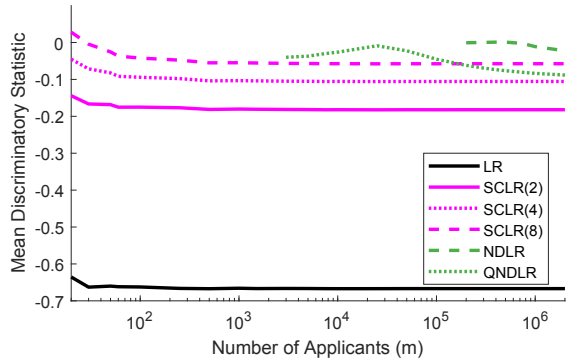
(a) The probability that a solution was returned for various training set sizes, m.



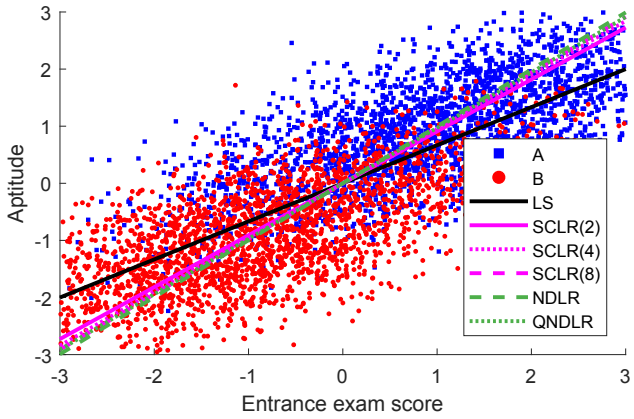
(b) The probability that each method will produce a solution that discriminates too much—a θ where $|d(\theta)| > \epsilon$—for various training set sizes, m.



(c) The MSE of the solutions produced by each method for various training set sizes, m.



(d) The mean discrimination statistic, $d(\theta)$, using the solutions produced by each method with various training set sizes, m.



(e) Examples of the lines found by several of the algorithms given $m = 500{,}000$ training points.

Fig. S16. Results of applying various linear regression algorithms to the illustrative example. All results are averaged over 200 trials. LR denotes ordinary least squares linear regression.

4.6 Application of NDLR and QNDLR to Real-World Data

The illustrative example provides an easily reproduced and understood example of how discriminatory behavior can manifest when using machine learning algorithms designed using the standard ML approach, and how it can be mitigated when using (quasi-)Seldonian algorithms. However, it is a simple synthetic example, which raises the question: does this sort of discriminatory (racist or sexist) behavior actually occur when using machine learning algorithms designed using the standard ML approach with real data? Recent research has suggested that the answer is affirmative: machine learning algorithms designed using the standard ML approach that were applied to important problems have acted in racist and sexist ways. For example, Datta et al. [105] showed that standard classification algorithms including logistic regression, support vector machines, and random forests can all produce racist and sexist behavior when used to predict whether or not a person is likely to commit a crime in the future. Similarly, Datta et al. [107] showed that Google's online advertising system is more likely to show advertisements for a high paying job to men than it is to show it to women, and Kay et al. [128] found that "image search results for occupations slightly exaggerate gender stereotypes and portray the minority gender for an occupational [sic] less professionally." This real-world discrimination is not limited to gender: Sweeney [113] found that Google AdSense was more likely to generate advertisements suggestive of an arrest record when searching for names associated with black people when compared to searches for names associated with white people. In another example, machine learning algorithms have been used to predict whether convicts will be likely to commit crimes in the future, and these machine predictions were considered during sentencing. A recent investigation into the behavior of these algorithms suggests that they may be twice as likely to *incorrectly* predict that a black person is likely to commit a crime than they are to incorrectly predict that a white person is likely to commit a crime [3].

In this section we present another example of how regression algorithms designed using the standard ML approach can be applied to real data, how this results in sexist behavior, and how (quasi-)Seldonian algorithms can be used to preclude sexist behavior. Whereas in our illustrative example we predicted the aptitude of job applicants based on their résumé, here we predict the aptitude of students applying to a university. Specifically, we use applicants' scores on nine exams taken as part of the application process to a university to predict what their *grade-point averages* (GPAs) will be during the first three semesters at university. Our training set consisted of data from 43,303 students. For each student, we are given three terms: X, Y, and T, as in the illustrative example. Here, $X \in \mathbb{R}^9$ is a vector of the student's 9 exam scores, Y is the student's mean GPA during the first three semesters of university (using the letter-to-number system $A = 4.0$, $B = 3.0$, $C = 2.0$, $D = 1.0$, and $F = 0.0$), and T is a binary value that indicates if the student is female or male.

For these experiments, we used *leave-one-out cross-validation*. That is, we predicted the performance of the i^{th} student after training using the data from all of the other students, and measured the resulting sample MSE and discrimination statistic when using various regression algorithms. We applied QNDLR and QNDLR(λ) using $\delta = 0.05$ and $\epsilon = 0.05$, as well as four algorithms designed using the standard ML approach: *least squares linear regression* (LS), an artificial neural network with ten neurons in its hidden layer (ANN) [129], a random forest (RF) [42], and *soft-constrained linear regression* (SCLR). For the first three

machine learning methods designed using the standard ML approach (LS, ANN, and RF) we used the standard implementations provided by MATLAB R2017b, and we implemented SCLR by performing gradient descent on the objective function provided in Eq. S7. We applied SCLR with $\lambda = 0.1$ and $\lambda = 1.0$, which we denote by SCLR(0.1) and SCLR(1.0) respectively, and we used these same values of λ when running QNDLR(λ).

Fig. S17 depicts the result of applying LS, ANN, RF, SCLR(0.1), SCLR(1.0), QNDLR, QNDLR(0.1), and QDNRL(1.0). In all trials (all 43,303 folds of leave-one-out cross-validation) QNDLR, QNDLR(0.1), and QNDLR(1.0) all always returned solutions (not NSF). Notice that the algorithms designed using the standard ML approach discriminate against female applicants (they produce large discrimination statistics), while QNDLR and QNDLR(λ) do not (they succesfully bound the discrimination statistic so that it is less that $\epsilon = 0.05$). The sample discrimination statistics (computed using leave-one-out cross-validation and rounded to two significant figures) when using the standard methods are -0.28 (LR), -0.27 (ANN), -0.27 (RF), -0.26 (SCLR(0.1)), and -0.04 (SCLR(1.0)), while the sample discrimination statistics of QNDLR, QNDLR(0.1), and QNDLR(1.0) are 0.03, 0.03, and 0.01, respectively. Although seemingly small numbers, the discrimination statistics for the standard methods (other than SCLR(1.0), which we discuss later) correspond to massive systematic discrimination against female applicants. To put into context the magnitude of a discrimination statistic of -0.27, consider Fig. S18, which provides a histogram of the GPAs of all 43,303 students, and notice that due to the clustering of GPAs towards the upper end of the spectrum, a difference of 0.27 is significant.

Notice that SCLR(λ) is capable of precluding significant discrimination when λ is properly tuned. Earlier we argued that selecting appropriate values for λ a priori (before observing the data from the problem at hand) is difficult. Notice that using $\lambda = 0.1$ results in significant discrimination against female applicants. Furthermore, the standard error bars in Fig. S17 indicate that even using $\lambda = 1.0$ resulted in solutions with discrimination statistics above 0.05 with probability greater than δ. By contrast, QNDLR(λ) combines a soft-constrained multiobjective approach with our framework. Even when using values for λ that would produce significant discrimination (e.g., $\lambda = 0.1$), QNDLR(λ) still ensures that the discrimination statistic will be below ϵ with probability at least $1 - \delta$. Also, notice that the mean prediction error for male applicants when using QNDLR(0.1) is above $\epsilon/2 = 0.025$. This does *not* mean that QNDLR(0.1) violates the behavioral constraint since the mean error for female applicants is not below $-\epsilon/2 = -0.025$. That is, the difference in mean prediction errors remains well below 0.05.

A common misconception about Fig. S17 is that it shows that QNDLR and QNDLR(λ) produced more accurate predictions for all students—they did not. To emphasize this point, in Fig. S19 we provide additional information about the estimators of GPA produced by each method. Notice that, just like in the illustrative example, the non-discriminatory algorithms tend to produce estimates with slightly higher MSE than the algorithms designed using the standard ML approach, but significantly less discrimination. This slightly higher MSE is the cost associated with precluding significant discrimination when applying a quasi-Seldonian algorithm for this application.

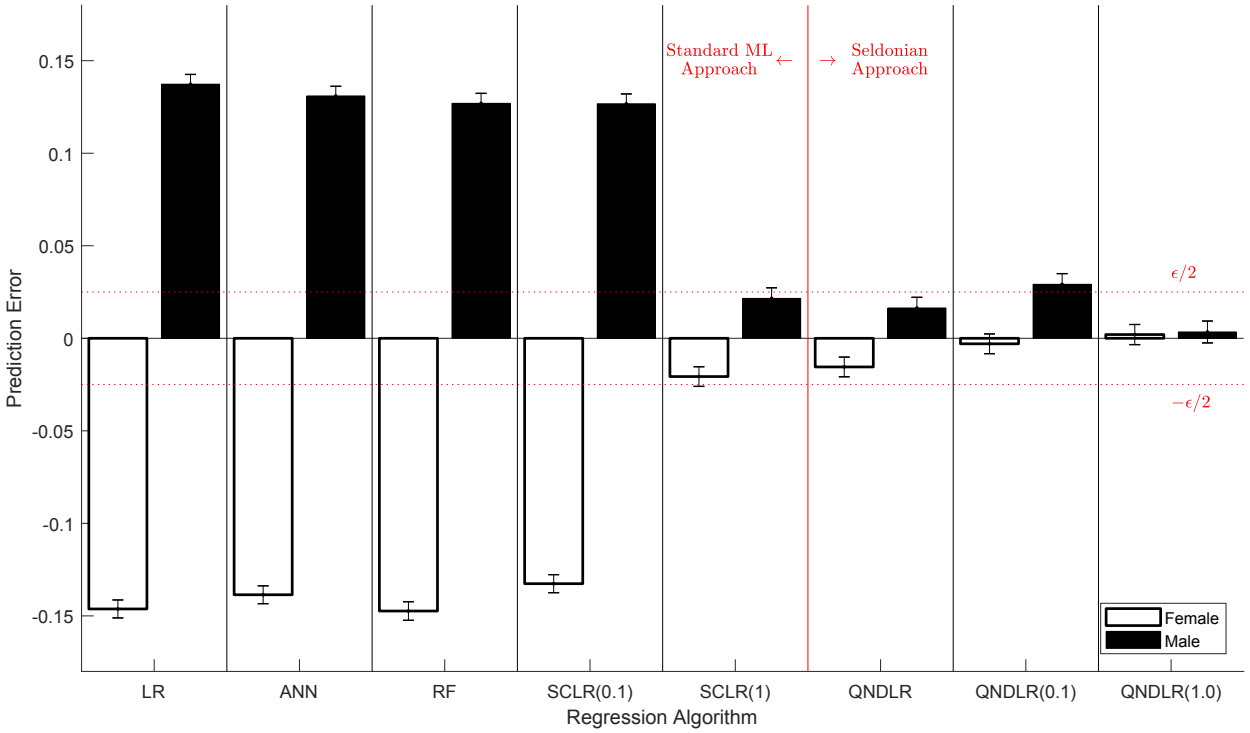


Fig. S17. Empirical results using various regression algorithms to predict student GPAs based on entrance exam scores. Results are averaged over all 43,303 folds when using leave-one-out cross-validation, and standard error bars are provided.

4.7 Application Using Other Definitions of Fairness

Since the Seldonian framework is a framework for designing machine learning algorithms, not a particular algorithm, it can be used to create classification algorithms. Furthermore, a primary benefit of Seldonian algorithms is that they are not tied to a specific definition of undesirable behavior—they allow the user to define what he or she considers to be undesirable behavior. In the context of ensuring fairness, this means that Seldonian algorithms allow the user of the algorithm to define the definition of fairness that is appropriate for the application at hand. This generality is critical because users may not use fair machine learning algorithms if the provided fairness guarantees do not align with the users' goals.

To provide supporting evidence for the claim that Seldonian algorithms can allow the user to specify a variety of different definitions of undesirable behavior, we used the algorithm in Fig. S15 (using CMA-ES [130] to solve for θ_c on line 4) to perform *classification* using multiple modern definitions of fairness defined by other researchers in recent related literature. The transition from regression to classification involves replacing the objective function f (MSE) with a classification loss function, and defining g to encode other definitions of fairness. Specifically, we use the indicator loss function as our objective: $f(\theta) := \Pr(\hat{y}(X, \theta) = Y)$. Because some definitions, such as disparate impact, cannot be estimated without bias, and the concentration inequalities we use require unbiased estimates, we construct high-confidence bounds for these definitions by first constructing suitable confidence intervals for the terms that they depend on (which we call *base variables*), and then combining these intervals

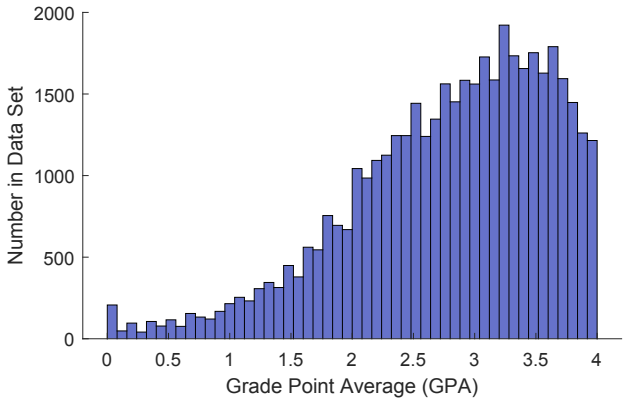


Fig. S18. Histogram of student GPAs in the data set.

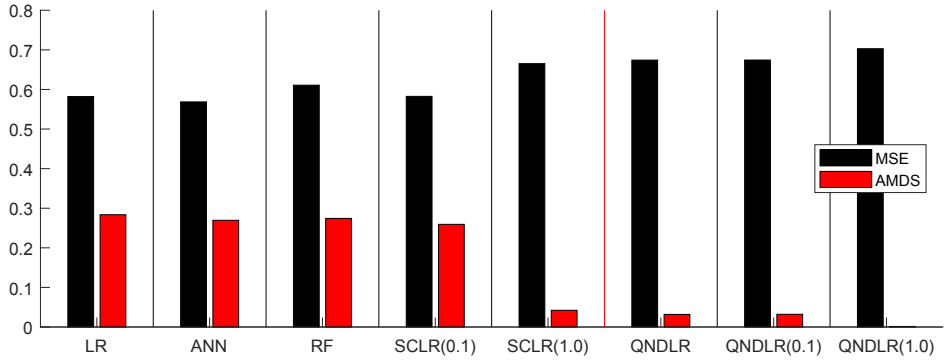


Fig. S19. The MSE and absolute mean discrimination statistic (AMDS)—the absolute value of the sample mean discrimination statistic over all trials—for the various regression algorithms.

into a confidence interval for g. For example, when computing the bound when g encodes disparate impact, we construct separate confidence intervals for $\Pr(\hat{y}(X,\theta){=}1|T{=}0)$ and $\Pr(\hat{y}(X,\theta){=}1|T{=}1)$, which we then use to bound g. Recall from our earlier discussion that we use the machine learning community's definitions of disparate impact and disparate treatment, which, at times, may not align with their legal counterparts.

To convert the GPA prediction problem into a classification problem, we used a GPA threshold of 3.0 to group students into "high performance" and "low performance" groups. We considered the task of predicting whether a student falls into the low- or high-performance group, while not discriminating based on sex. We defined $Y = -1$ to denote membership in the low-performance group (GPA $<$ 3.0) and $Y = +1$ to denote membership in the high-performance group (GPA $\geq$ 3.0). To demonstrate the generality of our approach, we repeated our experiment five times, each time using a different definition of discrimination: disparate impact, demographic parity, equal opportunity, equalized odds, and predictive equality. Because these definitions were originally formulated as hard constraints rather than real-valued metrics, we converted them to metrics by measuring the degree to which they are violated, and introduced a definition-specific tolerance parameter ϵ. For example, since demographic parity requires that the rate of positive predictions be equal for each type, we applied a constraint objective that measures the absolute difference between these rates, with

a tolerance ϵ_{DP}:

$$g_{DP}(\theta) \coloneqq |\Pr(\hat{y}(X,\theta)=+1|T=0) - \Pr(\hat{y}(X,\theta)=+1|T=1)| - \epsilon_{DP}. \tag{S32}$$

The tolerance for each definition will typically be determined by the application; for illustrative purposes, we used tolerances of -0.80, 0.15, 0.2, 0.35, and 0.2, respectively, for disparate impact, demographic parity, equal opportunity, equalized odds, and predictive equality.

For our experiments, we evaluate linear models trained using a variety of classification algorithms. We refer to our Seldonian approaches as *Seldonian classification* (SC) and *quasi-Seldonian classification* (QSC). To establish baseline performance, we compare to two standard classification algorithms and two algorithms that are specifically formulated to enforce certain definitions of fairness. First, we compare to linear models trained using stochastic gradient descent using the logistic loss, perceptron loss, and hinge loss, as well as logistic regression and linear support vector classification [131]. Because the performance of these baselines individually is not important to our results, we group these approaches together in our results, and refer to them as the *standard* approaches. While these methods are not designed to enforce fairness, we also evaluate two recently-proposed state-of-the-art fairness-aware algorithms, Fairlearn and Fairness Constraints.

As described previously, *Fairlearn* (FL) can enforce definitions of fairness that can be written as a set of linear constraints on conditional moments, such as confusion rates [24]. It includes a tolerance parameter that determines the maximum amount by which fairness can be violated, analogous to ϵ in our formulation. We used the implementation provided by Agarwal et al. [24], which includes code to enforce demographic parity and equalized odds. For other definitions of fairness, we set Fairlearn to enforce equalized odds as a surrogate fairness definition, and tested several settings of the tolerance parameter to assess the performance of the approach. When applying FL to definitions besides demographic parity and equalized odds, we evaluate FL using tolerance values of 0.01, 0.1, and 1.0.

In addition, we evaluate the approach of Zafar et al. [25], which we refer to as *Fairness Constraints* (FC). FC is designed to simultaneously enforce disparate treatment and disparate impact. However, because an objective based on disparate impact is non-convex and therefore difficult to optimize, FC uses a relaxed version that limits the covariance between the protected attributes and the model's predictions. Because there is not a simple correspondence between this covariance and the parameter p of disparate impact [25], we provide results for several parameter settings. As with FL, we evaluate performance using covariance parameter values of 0.01, 0.1, and 1.0.

When performing our evaluation, we trained each model to make predictions using non-sensitive features only—that is, all predictions were made using X, but not T. As a result, all of the models we evaluate satisfy the principle of disparate treatment. Also, notice that FC was only designed to limit disparate impact, FL was not designed to limit disparate impact, and the standard approaches were not designed to limit any of these definitions of fairness. Still, we report the performance of all of these methods for all definitions of fairness to show when they do and do not happen to successfully limit different definitions of fairness and to emphasize that only the (quasi-)Seldonian methods successfully ensure every type of fairness.

Fig. 3 (main text) presents results showing the behavior of these Seldonian and non-Seldonian algorithms when predicting the performance group for students while precluding

disparate impact (first row), demographic parity (second row), equal opportunity (third row), equalized odds (fourth row), and predictive equality (fifth row). For each algorithm, we evaluate how three metrics vary with the amount of data used for training: classification accuracy (left), solution rate (the probability with which each algorithm returns a solution, center), and failure rate (the frequency with which each algorithm violates the fairness constraint, right). The goals of varying the amount of training data are to expose the tendency for some algorithms to exhibit unfair behavior when smaller data sets are available and to show how little data our algorithm requires to frequently return solutions. For each position on the horizontal axis (number of training samples), we conducted 250 trials. For each trial, we resampled a training set with the specified number of training samples without replacement, and a larger data set without replacement from the remaining data points. We used this larger data set to evaluate the classifiers produced by the algorithms.

For all definitions of fairness, the failure rates for both our Seldonian and quasi-Seldonian classification algorithms are consistently below $100\delta\%$ and therefore satisfy the behavioral constraints. While other methods that consider fairness were also fair to use in some settings, each violated the behavioral constraints provided by our approach for some definitions and data set sizes. In particular, even when used with the fairness definitions for which they were designed, such as using FC to enforce disparate impact or FL to enforce demographic parity, these approaches often returned unfair solutions when trained on small data sets. The standard classification algorithms, which ignore fairness and optimize accuracy alone, typically return unfair solutions for all data set sizes. This evaluation shows that our approach is general enough to apply to a variety of meaningful fairness definitions, while remaining fair even when applied in configurations that cause other fairness-aware methods to fail.

The ability of our Seldonian algorithms to satisfy the behavioral constraints comes at a cost, reflected in Fig. 3 by the, in some cases, low accuracy and solution rates. This is not unexpected—it is unreasonable to assume that arbitrary fairness constraints can be enforced without a loss of accuracy—and when provided with sufficient data, SC and QSC consistently return fair solutions that achieve comparable accuracy to the best-performing alternative methods, which are often unfair.

It is important to note that SC and QSC are preliminary classification algorithms that fit within the Seldonian framework; given the performance of these algorithms, we consider the development of more sophisticated Seldonian methods to be an exciting direction for future work. In essence, SC and QSC show that Seldonian classification algorithms can be effectively applied to significant problems of interest, but they can be improved in several ways. For example, using concentration inequalities other than Hoeffding's, such as Maurer and Pontil's empirical Bernstein bound [132], could provide tighter confidence intervals, which would increase the rate with which our methods return solutions, especially when little data is available for training. Bounds derived from bootstrap confidence intervals [91], despite only holding approximately, could also be used to produce quasi-Seldonian classification algorithms that are more likely to return solutions and that achieve higher classification accuracy compared to QSC. Furthermore, there are several existing fairness-aware methods that provide upper bounds on the severity of unfair behavior which could be adapted to produce Seldonian algorithms that perform well for particular definitions of undesirable behavior [24]. Alternatively, improvements to the candidate selection process used by our methods could yield Seldonian classification algorithms that pass the safety test more often

while achieving higher classification accuracy. In particular, Seldonian algorithms that use improved methods for balancing the objectives of attaining low classification accuracy and being likely to pass the safety test could be much more data-efficient than SC and QSC. These directions for future research show the generality of the Seldonian framework, and highlight the fact that the results presented here should not be taken as limits of what the framework can achieve, but as a proof-of-concept of what is possible.

In summary, we have applied our Seldonian and quasi-Seldonian algorithms for regression and classification to enforcing a variety of behavioral constraints that capture different notions of fairness. These examples highlight that the design of Seldonian algorithms that function with a realistic amount of data is tractable (i.e., the theoretical guarantees are not impractical), and also show that Seldonian algorithms are general enough to constrain many types of undesirable behavior. In the following section, we show how a quasi-Seldonian algorithm can be used to solve a reinforcement learning problem. This application further highlights how Seldonian algorithms can provide an interface that enables a broad class of behavioral constraints and further emphasizes that the Seldonian framework is about precluding *all* types of undesirable behavior, not just unfair behavior.

5 Example: A Seldonian Reinforcement Learning Algorithm and its Application to Diabetes Treatment

So far we have focused on (quasi-)Seldonian algorithms that solve the relatively simple problem of fitting a line to data to provide an easily accessible and broadly interesting example. We now show that the ideas we have presented are not limited to this simple setting. Specifically, we propose a (quasi-)Seldonian batch *reinforcement learning* (RL) algorithm. RL algorithms allow machines to learn by trial and error without the need for an oracle or teacher to provide correct decisions for a set of training examples. RL has been applied to a variety of challenging problems [133, 134, 135]. For an introduction to RL, see the work of Sutton and Barto [47].

In our prior work we developed Seldonian and quasi-Seldonian batch RL algorithms [136, 137], but we had not yet developed the Seldonian optimization framework and therefore did not generalize our methods to handle a broad class of behavioral constraint functions. The methods presented in those publications only allow for the following single behavioral constraint: with high probability the performance (expected return) of the solution (policy) proposed by our algorithm should attain at least some user-specified baseline. Still, the algorithm, *safe policy improvement* (SPI), that we proposed could be extended to provide the user with the ability to select behavioral constraints from within a broad class of possible constraints.

Rather than directly extend our previous work, which would result in an algorithm similar to the one in Fig. S15, here we propose a new quasi-Seldonian algorithm that takes a non-standard, but not uncommon [138, 139, 140, 141], approach to RL: we propose an algorithm that searches the space of probability distributions over policies rather than an algorithm like SPI that directly searches the space of policies. This decision simplifies the use of importance sampling [142] by removing the product over time that usually appears when using importance sampling for RL [143]. It also makes it easier for the user to understand what policies our algorithm could feasibly deploy.

Let $\mathcal{P}$ be a set of (stochastic or deterministic) policies of interest for an episodic *Markov decision process* [144, MDP]. Each $p \in \mathcal{P}$ denotes one way that the agent could make decisions as a function of the input it receives. Let $\mathcal{H}$ be the set of possible outcomes that could occur during one episode of an MDP: each $h \in \mathcal{H}$ is a sequence of states (observations), actions (decisions), and rewards that describes the agent's interactions with its environment during one episode. We refer to each $h \in \mathcal{H}$ as the *history* of an episode. Each policy, $p \in \mathcal{P}$, induces a distribution over $\mathcal{H}$. We write $H \sim p$ to denote that the history-valued random variable H will be generated using the policy p. Let $r : \mathcal{H} \to \mathbb{R}$ be the *return function*, where $r(h) \in \mathbb{R}$ is the *return* of the history h, which is a measure of how "good" h is, with larger values being preferable. The goal in RL is typically to find an *optimal policy* $p^\star$: a policy that maximizes the expected return:

$$p^\star \in \arg\max_{p \in \mathcal{P}} \mathbf{E}[r(H)|H \sim p]. \tag{S33}$$

We modify this goal to be the following: find an optimal distribution over policies (within a feasible set of distributions over policies). This allows for the application of our algorithm to control problems where the user can specify an initial distribution over (possibly deterministic) policies based on prior beliefs about which policies will perform well. Let μ_θ be a distribution over $\mathcal{P}$ for all solutions, θ. We write $P \sim \mu_\theta$ to denote that the policy-valued random variable P will be sampled from μ_θ. The expected return when using solution θ can then be written as $\mathbf{E}[r(H)|P \sim \mu_\theta, H \sim P]$.

We assume that m policies $P_1, \ldots, P_m$ were sampled independently from some *behavior* distribution μ^b and we assume that each of these policies was used to generate a history, yielding m histories $H_1, \ldots, H_m$. This *historical data*, formally defined as the random variable $D = \{(H_j, P_j)\}_{j=1}^m$, will be the input to our algorithm. Notice that D is a random variable because each H_j and P_j is a random variable. As before, let $\mathcal{D}$ be the set of all possible historical data sets, D. We can now state our goal as an SOP:

$$\arg\max_{a \in \mathcal{A}} \mathbf{E}[r(H)|P \sim \mu_{a(D)}, H \sim P] \tag{S34}$$

$$\text{s.t. } \forall i \in \{1, \ldots, n\} \ \Pr(g_i(a(D)) \leq 0) \geq 1 - \delta_i, \tag{S35}$$

where the expectation is taken with respect to the MDP representing the user's task, and where $\mathcal{A}$ is the set of functions, $a \in \mathcal{A}$, where $a : \mathcal{D} \to \Theta$ (where Θ is the set of possible solutions, which we discuss later).

The algorithm that we propose is likely not an optimal algorithmic solution (especially given that we do not know the MDP representing the user's task), although it is (quasi-)Seldonian. However, it allows for a broad class of behavioral constraint functions and provides the user with an expressive language with which to define these constraint functions. Specifically, to specify the i^{th} behavioral constraint function g_i, the user must select an additional return function $r_i : \mathcal{H} \to \mathbb{R}$, which is used to implicitly define the behavioral constraint:

$$g_i(\theta) \coloneqq \mathbf{E}[r_i(H)|P \sim \mu^b, H \sim P] - \mathbf{E}[r_i(H)|P \sim \mu_\theta, H \sim P]. \tag{S36}$$

That is, our algorithm must ensure that with probability at least $1 - \delta_i$, it will not change the distribution over policies to one that decreases the expected return, computed using the return function r_i.

Notice that this means the algorithm's user need not know which policies could cause desirable or undesirable behavior. For example, if the user can only recognize when a history, $h \in \mathcal{H}$, has an undesirable outcome, then he/she could define $r_i(h) = -1$ if h is an undesirable outcome, and $r_i(h) = 0$ otherwise. This would specify that with probability at least $1 - \delta_i$, the returned distribution over policies should cause undesirable outcomes to occur with probably no more than it was under the behavior distribution.

We will focus on designing a quasi-Seldonian algorithm rather than a Seldonian algorithm. In our earlier work we found that Seldonian batch RL algorithms often require large amounts of data. Although practical when large amounts of data are available, as in problems involving "big-data" [145], for many problems it is difficult to obtain the needed amount of data. By contrast, quasi-Seldonian methods using Student's t-test or bootstrap confidence bounds produced solutions given relatively small amounts of historical data, while still providing useful information about the probability that the behavioral constraints will be satisfied [137]. We therefore focus here on a quasi-Seldonian algorithm that uses confidence intervals based on Student's t-test. However, the approach taken here can be extended to use other confidence bounds, and proper use of a confidence bound like those produced by Hoeffding's inequality could produce a Seldonain algorithm.

To simplify the algorithm that we propose here, we assume that the set of solutions is small and finite; that is, $\theta \in \{1, \ldots, l\}$ for some small l. This assumption allows us to avoid partitioning the training data into two sets, one of which is used to select a single candidate solution, and the other to determine whether this one solution satisfies the necessary probabilistic bounds. Instead, since l is small, we are able to use the union bound to ensure that our probabilistic bounds hold across all possible solutions simultaneously. This assumption means that the algorithm that we propose is viable mainly when the user has a few ideas about how the initial distribution, μ_b, might be improved. Again, SPI [26] (extended to allow for more general behavioral constraints) is an example of how Seldonian and quasi-Seldonian algorithms can be designed without this assumption (by partitioning the training data as described above).

The algorithm we propose has two steps. Unlike the (quasi-)Seldonian regression and classification algorithms presented above, which compute a single candidate solution from data, this algorithm assumes that l candidate solutions have been provided. During the first step, it uses *high confidence off-policy policy evaluation* (HCOPE) methods [136, 146] to test whether each of the l solutions satisfies the behavioral constraints. Since these tests use the same data set when testing each of the potential policies, to avoid the problem of multiple comparisons [147], the bounds are each constructed to hold with probability at least $1 - \delta/l$, so that all hold simultaneously with probability at least $1 - \delta$, by Boole's inequality. In the second step, the algorithm that we propose searches through the set of solutions that were deemed safe in the first step and returns the single solution that it predicts will have the best performance. If none of the first-step solutions are deemed safe, the algorithm returns No Solution Found.

We now describe these steps in more detail. First, for each of the l possible solutions, each of the n behavioral constraints, and each of the m trajectories of historical data, we use *importance sampling* [142, 148] to construct an unbiased estimate $\hat{\rho}_{i,j,k}$ (where $i \in \{1, \ldots, l\}$, $j \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, m\}$), of $\mathbf{E}[r_j(H)|P \sim \mu_i, H \sim P]$, where μ_i is shorthand for μ_{θ_i}. For importance sampling to provide unbiased estimates, we require the assumption that

$\text{supp}(\mu_i) \subseteq \text{supp}(\mu^b)$ for all $i \in \{1, \ldots, l\}$, which is standard in off-policy policy evaluation research [143, 136, 149]. However, if for some $i \in \{1, \ldots, l\}$, $\text{supp}(\mu_i) \neq \text{supp}(\mu^b)$, we can leverage our knowledge of μ^b and μ_i to improve the ordinary importance sampling estimates [146]. In our algorithm we use this modified importance sampling estimator to construct each $\hat{\rho}_{i,j,k}$ (in the event that this modified importance sampling estimator returns fewer than m estimators for each i, j pair, we select $k \in \{1, \ldots, m'\}$ for some $m' \leq m$).

Once $\hat{\rho}_{i,j,k}$ has been computed for all values of i, j, and k, we use Student's t-test to obtain high confidence approximate lower-bounds on $\mathbf{E}[r_j(H)|P \sim \mu_i, H \sim P\mu]$ for all $i \in \{1, \ldots, l\}$ and $j \in \{1, \ldots, n\}$. We also use Student's t-test to compute high confidence approximate upper-bounds β_j on $\mathbf{E}[r_j(H)|P \sim \mu^b, H \sim P\mu]$ for all $j \in \{1, \ldots, n\}$. We use $\delta_j/(l+1)$ so that the high-probability approximate upper- and lower-bounds for all solutions hold simultaneously. Next, our algorithm checks which solutions, that is, which values of $i \in \{1, \ldots, l\}$, produced high confidence approximate lower-bounds that were all above their respective baseline values β_j. Those solutions are deemed *safe* because our algorithm could return any of them and it would be quasi-Seldonian. If no solutions are deemed safe, the algorithm returns No Solution Found (NSF).

If at least one solution is deemed safe, then our algorithm searches within the safe set of solutions, for the solution that it predicts will produce the largest expected return on the MDP. The predictions of expected returns are once again constructed using the modified form of importance sampling. Pseudocode for our algorithm is provided in Fig. S20. For simplicity, this pseudocode assumes that each distribution over policies is a probability density function and uses Riemann integrals.

5.1 On the Ease of Using Our Quasi-Seldonian Reinforcement Learning Algorithm

Algorithms designed using our framework should be easier for a user to deploy responsibly than algorithms designed using the standard ML approach. Above we have shown how difficult it can be to include probabilistic constraints on the behavior of an algorithm designed using the standard ML approach. Here we point out that our quasi-Seldonian RL algorithm is a strong example of a new type of algorithm that makes it easy for the user to place probabilistic constraints on the algorithm's behavior.

To ensure with high probability that undesirable behavior does not occur, when using our algorithm the user does *not* need to perform additional data analysis, does *not* need to have training in statistics and data mining, and does *not* need to have detailed knowledge of the problem at hand (such as knowledge about the true transition or return functions of the relevant MDP). Instead, the user only needs to be able to recognize undesirable behavior to define $r_i(H)$, a measure of how much undesirable behavior occurred in the outcome H. Other than the auxiliary return functions r_i and their corresponding confidence levels δ_i, our algorithm has no additional hyper-parameters that the user must tune. As a result of these properties, our new algorithm can be applied more easily to a variety of batch RL problems for which the user can define what constitutes undesirable behavior without knowing which policies cause undesirable behavior.

Input: **1)** Data set $D = \{(H_j, P_j)\}_{j=1}^{m}$, **2)** behavior distribution μ^b, **3)** MDP return function $r : \mathcal{H} \to \mathbb{R}$, **4)** n behavioral constraint return functions $r_1, \ldots, r_n$, where each $r_i : \mathcal{H} \to \mathbb{R}$, **5)** n behavioral constraint confidence levels $\delta_1, \ldots, \delta_n$, where each $\delta_i \in (0, 1)$, and **6)** l candidate distributions over policies $\mu_1, \ldots, \mu_l$, where $\text{supp}(\mu_i) \subseteq \text{supp}(\mu^b)$ for all $i \in \{1, \ldots, l\}$.

Assumptions: This algorithm assumes that the sum of roughly m i.i.d. random variables (the importance weighted returns) is normally distributed.

/* Upper bound the expected returns if μ^b and each r_j were to be used. */

for $j = 1$ *to* n **do**

  $\beta_j \leftarrow \frac{1}{m}\sum_{k=1}^{m} r_j(H_k) + \frac{1}{\sqrt{m}}\,\text{stddev}([r_j(H_1), r_j(H_2), \ldots, r_j(H_m)])\,\text{tinv}(1 - \delta_j/(l+1), m-1)$;

/* Determine which solutions, $\theta \in \{1, \ldots, l\}$ are safe. Loop over each solution and test whether it should be removed from safe. */

safe $\leftarrow \{1, 2, \ldots, l\}$; // Set of integers.

for $i = 1$ *to* l **do**

  $c_i \leftarrow \int_{\text{supp}(\mu_i)} \mu^b(p)\,\mathrm{d}p$; // Scalar.

  /* Check whether the i^{th} solution satisfies the j^{th} behavioral constraint. */

  for $j = 1$ *to* n **do**

    /* Load $\hat{\rho}$ with all of the importance weighted returns. */

    Create empty list, $\hat{\rho}$, of floating point numbers;

    for $k = 1$ *to* m **do**

      if $\mu_i(P_k) \neq 0$ **then** append $c\frac{\mu_i(P_k)}{\mu^b(P_k)} r_j(H_k)$ to the list $\hat{\rho}$;

    /* Check whether the lower bound is less than the baseline, β_j, and if it is, then mark the i^{th} solution as unsafe. */

    if $\hat{\rho} = \emptyset$ **or** $\text{mean}(\hat{\rho}) - \frac{\text{stddev}(\hat{\rho})}{\sqrt{\text{length}(\hat{\rho})}}\,\text{tinv}(1 - \delta_j/(l+1), \text{length}(\hat{\rho}) - 1) < \beta_j$ **then**

      safe $\leftarrow$ safe $\setminus \{i\}$;

/* The set ''safe'' now holds the set of safe solutions. If it is empty, return NSF. If it is not empty, search for the solution predicted to maximize expected return. */

if *safe* $= \emptyset$ **then return** No Solution Found (NSF);

for *idx* $= 1$ *to* length(*safe*) **do**

  $i \leftarrow$ safe[idx]; // Scalar integer.

  curPerf $\leftarrow \left(c_i \sum_{k=1}^{m} \frac{\mu_i(P_k)}{\mu^b(P_k)} r(H_k)\right) / \left(\sum_{k=1}^{m} \mathbf{1}_{(\mu_i(P_k)\neq 0)}\right)$; // Scalar.

  if *idx* $= 1$ **or** *curPerf* $>$ *bestPerf* **then**

    bestPerf $\leftarrow$ curPerf; // Scalar.

    bestIdx $\leftarrow$ idx; // Scalar integer.

return safe[bestIdx];

Fig. S20: Quasi-Seldonian Reinforcement Learning Algorithm.

5.2 Application of Quasi-Seldonian Reinforcement Learning Algorithm to Diabetes Treatment

We applied the quasi-Seldonian RL algorithm in Fig. S20 to a simplified simulation of providing personalized improvements to bolus insulin dosing for people with type 1 diabetes. This is an example of an important application where safe machine learning methods can complement existing approaches. While we use a fairly detailed metabolic simulator [32, T1DMS Version 3.2], this example is intended only as an illustration of the potential applications of our proposed framework and methods. The simulator is not a perfect model, and we consider

only a simple set of dosage recommendation policies. In Section 5.4, titled *Additional Considerations for Clinical Applications*, we discuss additional considerations that should be made prior to a real clinical application.

Often machine learning algorithms are evaluated using real-world examples to show the viability of the algorithm. For example, Grabczewski and Duch [20] were some of the first machine learning researchers to use the popular Wisconsin breast cancer data set [21] when evaluating a machine learning algorithm. These experimental results validate the proposed algorithm, but do not imply that the resulting classifier should be used by medical professionals. Rather, the papers by Grabczewski and Duch [20] and subsequent machine learning researchers provided tools that researchers with the appropriate domain knowledge and medical expertise could apply properly to cancer treatment [22, 150, 151, 152].

Similarly, in this section we show how a Seldonian RL algorithm could be applied to an important medical problem: optimization of insulin dosing for type 1 diabetes treatment. This experiment shows that the design of (quasi-)Seldonian RL algorithms is tractable and that these algorithms do not require an impractical amount of data when applied to realistic problems. Furthermore, this application presents a clear example of how (quasi-)Seldonian reinforcement algorithms can enforce safety guarantees that standard RL algorithms would violate. Just as Grabczewski and Duch [20] did not intend for their results on the breast cancer data set to result in the direct application of their classifier, these experiments should not be misinterpreted as a proposal for our quasi-Seldonian RL algorithm (with the return function, policy representation, and behavioral constraints that we select) to be deployed as-is to diabetes treatment.

Briefly, type 1 diabetes is the condition in which one's body does not produce sufficient insulin, a hormone that promotes uptake of glucose from the blood into muscle and liver, which lowers the blood glucose concentration. People with untreated diabetes tend to have high blood glucose levels, a condition called *hyperglycemia*, which can have significant negative consequences [153]. One treatment for diabetes is the subcutaneous injection of insulin using multiple daily injections with a syringe or an insulin pen, or a continuous infusion using an insulin pump.

If too much insulin is injected, blood glucose levels can become too low, a condition called *hypoglycemia*. Controlling hyperglycemia is important to prevent the long-term consequences of diabetes, and hypoglycemia is a common severe unintended consequence. Hypoglycemia can cause symptoms ranging from palpitations, sweating, and hunger, to altered mental status, confusion, coma, and even death [33, 34, 69]. Moreover, severe instances of hypoglycemia can triple the five-year mortality rate [69], and recurrent moderate hypoglycemia causes brain adaptations that impair a person's ability to detect future potentially severe instances of hypoglycemia [35]. Such hypoglycemia unawareness is clinically important because it impairs a person's ability to make informed decisions prior to a hypoglycemic episode, such as whether it is safe to drive.

Due to the severity of poorly regulated blood glucose levels, a critical decision is how much insulin a person should inject to mitigate hyperglycemia without inducing hypoglycemia. In addition to *basal insulin*, which regulates blood glucose between meals, an important challenge is to determine dosages of *bolus insulin*, which counteracts the increase in blood glucose that results from eating meals. A *bolus calculator*, which can range from software within an insulin pump to a cell phone application, tells a person how much bolus insulin

they should inject prior to eating a meal. Here, we show how our Seldonian RL algorithm can be applied to personalizing the parameters of a bolus calculator. While the basal dosing may or may not also be adapted in a clinical application [154], in our simulations the basal dose is held constant.

Since the amount of bolus insulin to inject depends on many factors, there has been significant interest in adaptive or intelligent bolus calculators, particularly using control theoretic methods, that can use data from the outcomes of previous injections to adapt treatments to each patient. Examples include model predictive control algorithms [155, 156, 157, 158], proportional derivative controllers [159, 160, 161], proportional integral derivative controllers [162, 163, 164], and Mamdani type fuzzy logic controllers [165]. These control algorithms, as well as other intelligent insulin delivery systems, have seen strong clinical success in deployed systems such as the Medtronic 670G, Space GlucoseControl system, and zone-MPC, among others [166, 167, 168, 169, 170, 171, 172]. For a history of bolus calculators, see the work of Schmidt and Nørgaard [31], and for a review of early insulin delivery controllers, see the work of Hovorka [173].

Another notable control-based approach to calculation of insulin dosage is the *Run-to-Run* (R2R) approach first proposed by Sachs et al. [174]. R2R is a control approach related to iterative learning control and repetitive control [175] wherein a controller is used to control a system for a sequence of runs (cf. RL's episodes). In the context of an adaptive bolus calculator, each run corresponds to a day (24 hours), and each day is broken into a set of segments. The amount of insulin given during each segment is based on a performance measure calculated for that segment of the previous day. R2R controllers for insulin dosing have promising stability properties under mild linearity assumptions [176, 177] and have obtained promising empirical results both *in silico* [178, 179, 180, 176, 181, 177, 182, 180, 183, 184, 37, 185] and clinically [186]. Another recent approach combines adaptive updating of the insulin parameters with case-based reasoning [187], showing promising preliminary clinical results [188].

These existing methods often use a patient dynamics model that includes a gain parameter and patient-specific values for the target maximum and minimum blood glucose values at certain points during the day. This gain parameter is typically either estimated for the population or requires tuning for individuals based on existing data. Our proposed reinforcement learing approach is complementary to these approaches, as it could be used to improve the gain for a particular patient based on that patient's own data, and in a way that provides guarantees on the adaptive bolus calculator's performance for that individual. In addition, while existing R2R algorithms work well if the gain is well specified and the proposed target glucose levels are achievable, it is unclear what their behavior will be if it is not possible to achieve the desired target glucose levels for a particular patient (e.g., will the resulting behavior satisfy one target value but not the other, or will some other behavior result). In contrast, our framework allows a medical expert to specify a constraint on the desired behavior, such as that the insulin dosage policy may not be altered in a way that increases hypoglycemia relative to the current policy for this particular patient.

Despite the potential benefits of using supervised learning [189] or RL algorithms to create personalized adaptive bolus calculators [30] or adaptive basal and bolus therapies [190, 191], their use has not yet seen the successes of other approaches. This may be in part because these existing approaches have used RL algorithms designed using the standard ML approach and which therefore lack meaningful safety guarantees. Whereas the stability properties of

conventional control algorithms are well understood, most RL algorithms have only asymptotic convergence guarantees [192] and in practice are sensitive to the setting of hyperparameters like step sizes, exploration rates, trace decay rates, policy representations, and value function representations [47]. So, although RL algorithms *attempt* to maximize the expected return, as a consequence of the random nature of data, or due to imperfectly tuned hyperparameters, in practice they often return policies that decrease the expected return both during and after learning.

One might wonder if one could regulate undesirable behavior by changing the reward function to further penalize undesirable behavior. Because RL algorithms can return policies that decrease the expected return relative to the initial policy, this is not sufficient to ensure that undesirable behavior will not occur. Furthermore, it is often unclear how much to penalize an undesirable outcome, such as hypoglycemia, to ensure that it is avoided, while still ensuring that the approach optimizes a desired primary objective, for example, minimizing the frequency of hyperglycemia. Existing RL algorithms do not provide a straightforward mechanism to allow users to insert safety constraints on behavior that are separate from the primary objective (maximizing the expected return). For example, they do not provide a means for ensuring that the policy (or distribution over policies) will not be changed in a way that increases the prevalence of low blood glucose for a particular patient (increases the prevalence of undesirable behavior), while increasing the primary objective function that trades off hypoglycemia and hyperglycemia (increasing the expected return). Thus, as an application of machine learning (reinforcement learning) held back by a lack of safety guarantees, creation of a personalized adaptive bolus calculator presents a clear example of the benefits of Seldonian RL algorithms over standard RL algorithms.

5.2.1 Simulation Design

We now describe our experimental setup for comparing Seldonian RL algorithms with classical RL algorithms for simulated adaptive bolus calculation. Our aim is to illustrate how a domain expert might use an algorithm designed using our new framework to improve the policy (controller) for a particular patient, while ensuring that, with high probability, the controller will not be modified in a way that violates safety constraints specified by the domain expert. We focus on safety constraints pertaining to hypoglycemia due to the large negative clinical implications of this condition, but our approach is applicable to many alternate constraints, as we illustrate below.

To simulate a patient we used a newer version of the simulator used by Bastani [30]: Version 3.2 of the *type 1 diabetes metabolic simulator* (T1DMS) [32]. T1DMS is a metabolic simulator that has been approved by the US Food and Drug Administration as a substitute for animal trials in pre-clinical testing of treatment policies for type 1 diabetes, and is often used to evaluate adaptive bolus calculators [176, 181, 182]. For an initial illustration we selected subject adult#003 within T1DMS.

One strength of our approach is that it provides per-subject guarantees. That is, standard ML approaches to constructing and evaluating adaptive bolus calculators often measure performance across a population of individuals and argue that new bolus calculators work better for the population (with arguments of statistical significance over the entire population). These arguments of statistical significance provide a form of safety guarantee about the

performance of the new adaptive bolus calculator for the *population*. We instead focus on providing a personalized guarantee—that the controller for an individual subject will not be changed in a way that is worse for that one individual. We therefore perform multiple simulations of a particular individual, with each simulation using different random meal times and random samples within the learning algorithm. Although initially we focus on the individual adult#003 within T1DMS, later we show that this individualized approach is effective for personalizing treatments for all ten simulated adults within T1DMS.

Following the experimental setup proposed by Bastani [30], the subject is provided with three meals of randomized sizes at randomized times. Due to these random meal sizes and times, applying the same policy parameters for two different days can result in different outcomes. Also following previous work, we largely adopt the experimental design proposed by Bastani [30], which adapts a relatively simple type of bolus calculator that does not include a dependence on *insulin on board*—a statistic that tracks how much insulin has been injected previously. In this experimental setup, the amount of insulin (measured using *insulin units*) that a person with diabetes is instructed to inject prior to eating a meal is given by:

$$\text{injection} = \frac{\text{blood glucose} - \text{target blood glucose}}{CF} + \frac{\text{carbohydrate content of meal}}{CR}, \quad \text{(S37)}$$

where blood glucose is an estimate of the person's current blood glucose (measured from a blood sample and using milligrams per deciliter of blood, i.e., mg/dL), target blood glucose is the desired blood glucose (also measured using mg/dL), which is typically specified by a physician, the carbohydrate content of meal is an estimate of the size of the meal to be eaten (measured in grams of carbohydrates), and CR and CF are two real-valued parameters that, for each person, must be tuned to make the treatment policy effective. CR is the *carbohydrate-to-insulin ratio*, and is measured in grams of carbohydrates per insulin unit, while CF is called the *correction factor* or *insulin sensitivity*, and is measured in insulin units per mg/dL. Following the prior work on which we based our experimental design [30], where a target blood glucose of 6 mmol/L was used, we select a target blood glucose of 108 mg/dL (a close approximation to 6 mmol/L).

In the RL context, CR and CF are the parameters of a parameterized policy for an MDP in which actions correspond to recommended injection sizes and the state is a complete description of the subject at the current time.[4] The parameterized policy uses function approximation—it depends only on an observation (the blood glucose estimate acquired from a blood sample) that depends on the current state of the person.[5]

Intuitively, the return function should penalize deviations of blood glucose from optimum levels (with larger penalties for blood glucose levels that are too low). Precisely defining a return function of this sort is difficult because there are two conflicting goals: **1)** keep blood glucose levels from becoming too high, and **2)** keep blood glucose levels from becoming too low. Because hypoglycemia can have more severe consequences than hyperglycemia, the goal

[4]Note that the complete state of the subject is not likely to be observable, but as we will define next, the policy will only depend on an available observation, even though the dynamics underlying how insulin doses impact a person may be much more complex and depend on other aspects of the patient's health and context.

[5]Alternatively, this problem could be modeled as a *partially observable Markov decision process* (POMDP). However, since we focus on *state-free policies* [193, Section 7], the MDP framework with function approximation is sufficient.

is typically to minimize the time that a person is hyperglycemic, subject to the constraint that hypoglycemia never occurs. In practice, however, hypoglycemia cannot be completely precluded [69]. This means that the return function must be selected to trade-off between hyperglycemia and hypoglycemia, a problem described in detail by Bastani [30, Section 1.3.6].

Here the user of our quasi-Seldonian RL algorithm (Fig. S20) would have to make a decision: what return function quantifies their personal view of the trade-off between hypoglycemia and hyperglycemia, and what auxiliary return functions capture their view of undesirable behavior? Because our aim is to evaluate our Seldonian approach, not to advocate a particular answer to these questions, we adopt the return function used in previous work [30]. To this end, we let the history from each day contain a record of blood glucose levels at each minute, i.e., $h = (g_0, g_1, g_2, \ldots, g_{1440})$, where g_t denotes the blood glucose measurement t minutes after midnight, in mg/dL. The return function is then given by:

$$r(h) \coloneqq \frac{1}{1441} \sum_{t=0}^{1440} \begin{cases} -\frac{(g_t - 108)^2}{1623} & \text{if } g_t < 108\,\text{mg/dL} \\ -\frac{(g_t - 108)^2}{3246} & \text{if } g_t \geq 108\,\text{mg/dL}. \end{cases} \tag{S38}$$

These seemingly peculiar constants arise because Bastani [30] presented their reward function using mmol/L rather than mg/DL. This return function effectively computes a *reward* at each minute of the day, and the return for the day is then the average reward during the day.

The second decision that the user of our quasi-Seldonian RL algorithm must make is how to define the behavioral constraints through the auxiliary return function. Again, because our goal is not to promote a particular mapping of the diabetes management problem to the Seldonian framework, here we present one possible definition of the auxiliary return function r_1, and later we present results using this and other potential definitions that could capture the safety restrictions desired by different diabetes management experts. Precisely, we define an r_1 that penalizes low blood glucose levels:

$$r_1(h) \coloneqq \begin{cases} \frac{1}{1441} \sum_{t=0}^{1440} -\frac{(g_t - 108)^2}{1623} & \text{if } g_t < 108\,\text{mg/dL} \\ 0 & \text{if } g_t \geq 108\,\text{mg/dL}. \end{cases} \tag{S39}$$

Setting the policy parameters CR and CF to specify the insulin dosage policy that yields the best results for an individual based only on basic knowledge about a person (age, weight, etc.) is difficult, and perhaps impossible. Here we consider the case in which a physician initially proposes ranges of possible values for CR and CF for a particular individual. The parameter values for any insulin dosage policy considered should lie within these ranges. We refer to $\mathbf{E}[r_1(H)]$ as the *prevalence of low blood glucose* hereafter. Below we also discuss the mean-time-hypoglycemic-per-day, a different possible measure of the prevalence of low blood glucose.

Our aim is to evaluate how a batch RL algorithm can take in a series of previously applied policies (settings of CR and CF) for a patient, collected across $m \in \mathbb{N}_{>0}$ days, and output a new distribution over policies that, *for this particular patient*, increases the expected return as defined by Eq. S38. Furthermore, this algorithm must ensure that the safety constraint is satisfied; that is, it must ensure that with high probability the distribution over policies will not be changed in a way that increases the prevalence of low blood glucose. As an additional safety measure, we might also require the hard constraint that the RL algorithm

will never deploy values of CR and CF outside the range specified by the physician. This application therefore combines aspects of a multiobjective problem (the return function trades-off hypoglycemia and hyperglycemia), a problem with hard constraints (that the values for CR and CF will always be within the window specified by the physician), and the high-probability behavioral constraints allowed by the Seldonian framework (that the prevalence of low blood glucose will not be increased).

Batch RL algorithms for this problem can proceed by taking in all prior data for that person for each day, and evaluating it to determine if it is possible to output a better policy for that individual. For simplicity we assume that data is gathered from values of CR and CF sampled from a uniform distribution μ^b over the specified input range of CR and CF for that individual, and each pair of policy parameters is used for one day. Note that some amount of variability over time in the policy parameters is essential for any improvement to be possible. If only a single pair of CR and CF parameters are used at all times, our algorithm will not be able to evaluate the potential performance of any other policy. In RL this variability is referred to as exploration. Although here we consider a simple form of exploration to illustrate our method, our approach can be combined with other methods of providing variable insulin dosage recommendations, including stochastic adaptive policies such as those considered in prior RL applications in this setting [30], or with other adaptive bolus calculator methods that vary the insulin dosage policy parameters over time.

These desired properties of a batch policy search algorithm are captured by the Seldonian optimization problem presented in Eq. S34, where the set of algorithms, $\mathcal{A}$, is restricted to contain only those algorithms that will never return values of CR and CF outside a specified input range, and where there is a single behavioral constraint ($n = 1$), with auxiliary return function r_1, as defined in Eq. S39. That is, a (quasi-)Seldonian algorithm must ensure that with high probability the prevalence of low blood glucose (measured using r_1) will not increase, and should try to maximize the expected return (using the MDP return function) otherwise. Because safety is critical, we selected a small constraint confidence level: $\delta_1 = 0.05$.

Our quasi-Seldonian RL algorithm (Fig. S20) is a viable quasi-Seldonian algorithm for this problem, which allows for the specification of a set of possible distributions over policies that will be considered. We selected 27 such distributions, each of which is a uniform distribution over ranges for CR and CF that are subsets of the support of μ^b. Furthermore, each of the 27 distributions contains 1/4 the support of μ^b. The 27 distributions were generated by an automatic tiling scheme to sample various ratios of the range of CR to the range of CF within the support of μ^b. Examples of the ranges of three of these 27 distributions are provided in Fig. S23 as blue, black, and white boxes.

We modified the algorithm in Fig. S20 to include a model-based control variate in the importance sampling estimate [194, 149, 195]. The approximate model used to construct the control variate was a different setting of T1DMS that produces similar responses to adult#003, but which is not the same. Importantly, the optimal values of CR and CF under the approximate model cause frequent episodes of hypoglycemia: the model cannot be trusted to select values for CR and CF. However, the approximate model does provide an effective control variate, which decreases the variance of importance sampling estimates. Furthermore, we observed similar, although not quite as strong, results without the use of this model-based control variate.

5.2.2 Specifying the Initial Input Range of Policy Parameters.

For our *in silico* subject (adult#003), we assume that the physician specifies the initial ranges of CR and CF to be the intervals $[8.5, 11]$ and $[10, 15]$ respectively, as shown by the blue box in Fig. S21. That is, the initial distribution over policies, μ^b, is the uniform continuous distribution over this range. Fig. S23 provides a zoomed in view of the objective function over this range. Notice that this range contains near optimal policies, where $CR \approx 10$, as well as some undesirable policies, e.g., when $CR \approx 8.5$. Furthermore, notice that we observe the same trend as Bastani [30]: when CR is selected properly, performance is robust to the value of CF. We selected this range after performing a broader computation of the expected returns (undiscounted sums of rewards) that results from using either the MDP return function or the auxiliary return function, and using various values of CR and CF within a reasonable range, $CR \in [5, 50]$ and $CF \in [1, 31]$. These estimates are depicted in Fig. S21. When CR is too small, hypoglycemia occurs often: the mean returns using the auxiliary return function or MDP return function are both large negative values. When CR is too large, instances of hypoglycemia decrease (the mean return using the auxiliary return function plateaus at zero), but instances of hyperglycemia increase (the mean return using the MDP return function decreases). Thus, the goal of an algorithm is to identify values for CR and CF that lie along the ridge of the objective function that occurs around $CR \approx 10$, erring towards values of CR that are too large.

Notice that these plots are typically *not* available to the physician selecting initial values for CR and CF, nor are they available to an algorithm that might try to improve upon a physician's initial educated guess as to what values of CR and CF will work for the subject. Furthermore, it is difficult for any agent (a physician or RL agent) to accurately estimate these plots from data: each plot is the result of 44,000 simulations. To visualize the difficulty of estimating the expected return for a single CR and CF pair, consider Fig. S22, which shows the result of applying two different CR and CF pairs, one that lies just beyond the ridge resulting in frequent hypoglycemic episodes, and the other that is near-optimal and lies near the top of the ridge. Notice the high variance of the returns relative to the differences between the expected returns using different values of CR and CF (cf. Fig. S21). This high variance means that it is difficult to determine from small amounts of data which of these two CR and CF pairs is better, let alone search the uncountably infinite space of CR and CF pairs for their optimal settings. Thus, one might expect that the high confidence guarantees required of (quasi-)Seldonian algorithms might be impractical. Our results show that quasi-Seldonian algorithms can be effective even for this challenging problem.

5.2.3 Experimental Results and Discussion

We applied our quasi-Seldonian RL algorithm (Fig. S20) and a non-Seldonian algorithm, each for 200 trials. The non-Seldonian algorithm used importance sampling (with the model-based control variate) to estimate the expected return for each of the 27 possible policy distributions used, and output the distribution that it predicted will result in the largest return (in terms of the MDP return function). This corresponds to our quasi-Seldonian RL algorithm, but without any behavioral constraints.

Fig. S24 (left) shows that the quasi-Seldonian algorithm was able to return solutions

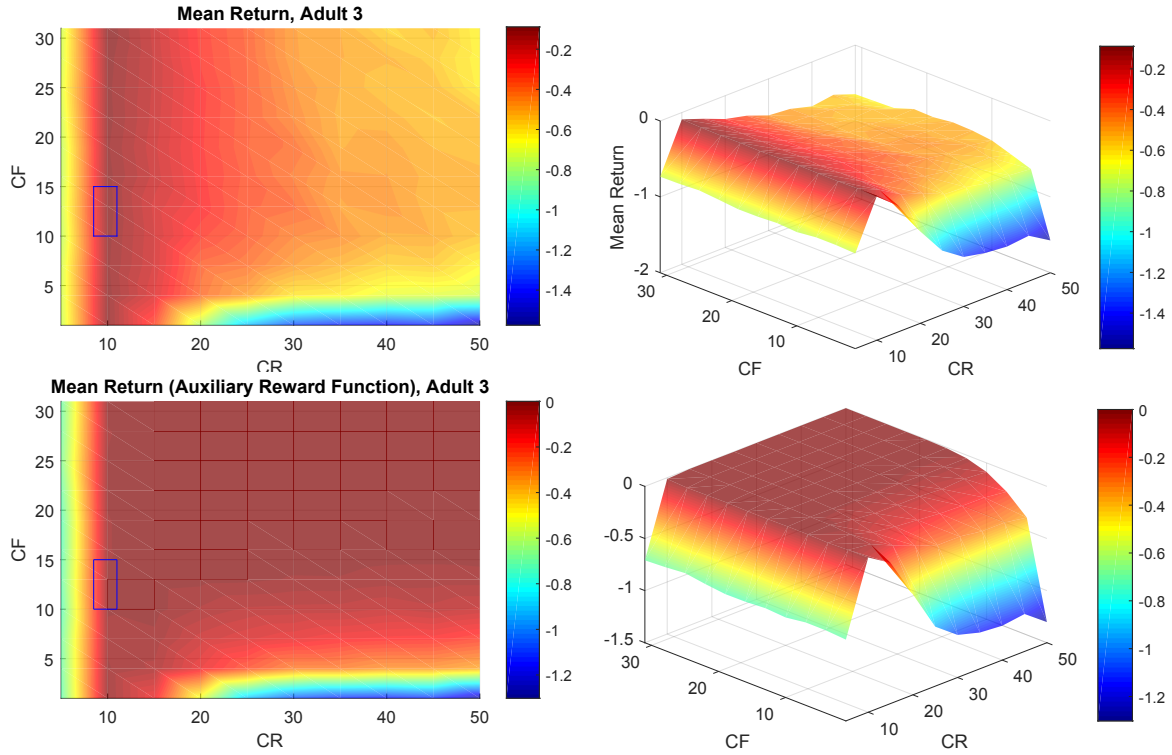


Fig. S21. The mean returns using the reward function of the MDP, which penalizes both hyperglycemia and hypoglycemia (top row), and the mean returns using the auxiliary reward function r_1, which only penalizes hypoglycemia (bottom row). The blue box in the plots in the left column depict an initial range of CR and CF values that might be selected by a physician. The values of CR were tested at intervals of 5 while the values of CF were tested at intervals of 3. Each CR and CF pair was evaluated using Monte Carlo sampling—500 days worth of data. Values between the grid of sampled points are interpolated using linear interpolation.

(other than NSF, in which case the physician's policy would not be changed) given as little as one month of data. Given 5 months of data (roughly 180 days), the quasi-Seldonian algorithm always returned a new distribution for CR and CF that was different from the initial distribution.[6] While the quasi-Seldonian algorithm sometimes did not return a new policy distribution, the RL algorithm designed using the standard setting always did, but at the cost of often deploying policies that result in increased prevalence of low blood glucose, as shown in Fig. S24 (right). By contrast, across all trials, the quasi-Seldonian algorithm never returned a distribution over policies that increased the prevalence of low blood glucose—much less than the 5% limit required of the algorithm.[7] That is, the algorithm designed using the standard ML approach would not be safe to deploy for adult#003, unlike the quasi-Seldonian

[6]Recall from earlier that each plot in Fig. S23 was generated using a large number of simulations. This large number of simulations was to illustrate how different CR and CF values perform, while Fig. S24 shows the performance of our algorithm.

[7]Without the control variate, the quasi-Seldonian algorithm returned distributions over policies that increased the prevalence of low blood glucose more frequently, but still with probability well below 0.05.

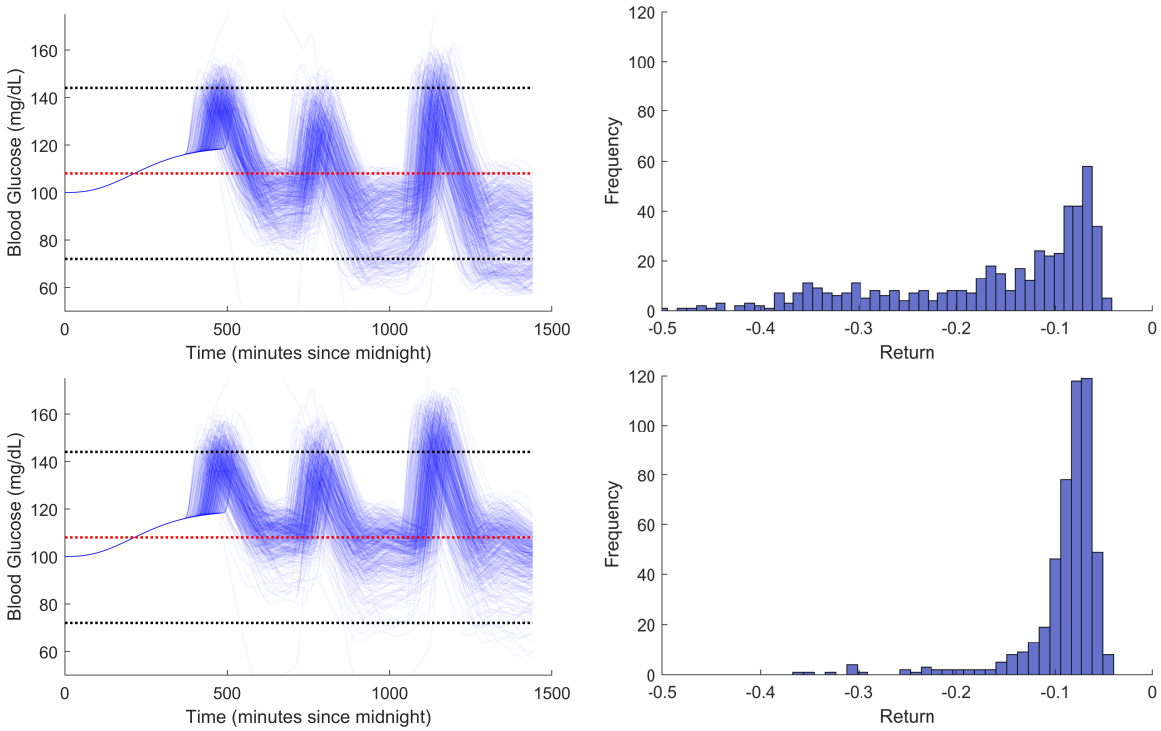


Fig. S22. Examples of 300 different days using $CR = 8.5$, $CF = 10$ (top row) and $CR = 10$, $CF = 12.5$ (bottom row). The former results in frequent hypoglycemia, while the latter is a near optimal policy. The plots on the left show the blood glucose throughout the day, with the black lines denoting a desirable range of blood glucose levels [30], and the red line denoting optimal levels. The three increases in blood glucose correspond to the times when the subject eats breakfast, lunch, and dinner. The plots on the right show histograms of the resulting returns (sum of rewards) computed using the MDP reward function (returns less than -0.5 occur but are not shown).

algorithm, which meets the specified safety constraints.

Notice also that Fig. S24 shows that the costs associated with providing high probability safety guarantees are relatively minor. Specifically, given roughly 75 days of data, these results suggest that with probability at least 95%, the non-Seldonian algorithm returned policies that would not increase the prevalence of low blood glucose (when applied to adult#003). Similarly, the quasi-Seldonian algorithm began returning solutions frequently given roughly 75 days of data. This lack of significant lag between when the non-Seldonian algorithm began to act in a safe manner and when our quasi-Seldonian algorithm began returning solutions (when it was able to automatically determine that returning solutions would be safe) shows that there is little cost associated with requiring a safety guarantee in this case. If the quasi-Seldonian algorithm were not data efficient, then it would not begin returning solutions frequently until long after the non-Seldonian algorithm tended to return solutions that do not increase the prevalence of low blood glucose.

Fig. S25 presents box plots of the expected returns of the policies produced by the Seldonian and non-Seldonian algorithms when run using different amounts of data. To obtain the box in the left plot at the horizontal mark at 50, the non-Seldonian algorithm was trained

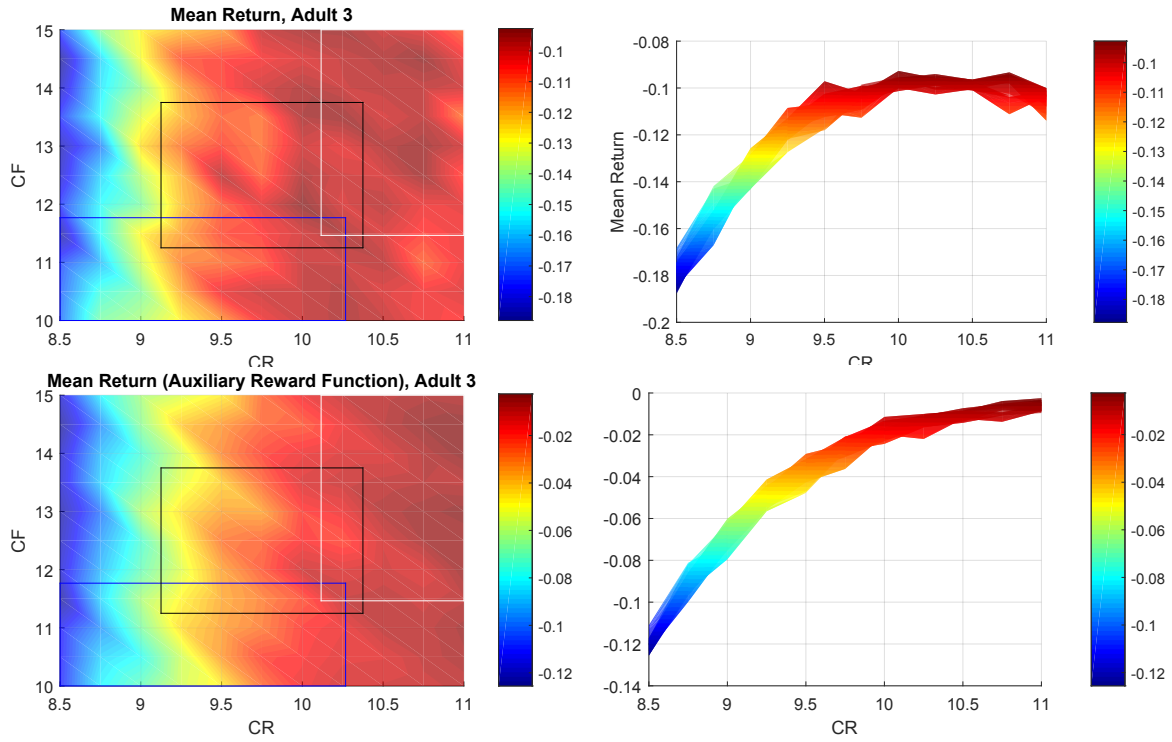


Fig. S23. A zoomed in view of Fig. S21. The boundaries of these plots are the range of CR and CF used by the initial distribution over policy parameters, μ^b. The plots on the right show a side view, which shows the difference between the reward function of the MDP (which penalizes both hyperglycemia and hypoglycemia) and the auxiliary reward function, r_1, which only punishes hypoglycemia. For all of these plots, CR was varied by increments of 0.25 and CF was varied by increments of 0.5, each CR and CF pair was evaluated using Monte Carlo sampling with 500 days of data, and values between sampled CR and CF values are interpolated using linear interpolation. The blue, black, and white boxes in the plots on the left are discussed later in the text.

using data collected from 50 days. The policy distribution that it returned was then evaluated for 500 simulated days to estimate the expected return that would result if it were to be used. We repeated this process 200 times, and the distribution of the resulting expected return estimates from these 200 trials is depicted by the box at the 50-mark on the horizontal axis. The left plot, which presents results for the non-Seldonian algorithm, shows that with small amounts of data, policies that were worse than the initial policy distribution were sometimes returned. This shows why modifying the rewards or returns to punish hypoglycemia more is not a solution because the algorithm does not always increase the expected return. By contrast, the Seldonian algorithm always increased the expected return (even though, in this case, the behavioral constraint only required improvement with respect to the expected auxiliary return). This plot therefore shows that the Seldonian algorithm succeeded at optimizing the primary objective, while Fig. S24 shows that it did so subject to the behavioral constraint.

In this study the quasi-Seldonian algorithm was provided with a range of admissible values for CR and CF for adult#003, i.e., $CR \in [8.5, 11]$ and $CF \in [10, 15]$, which is a

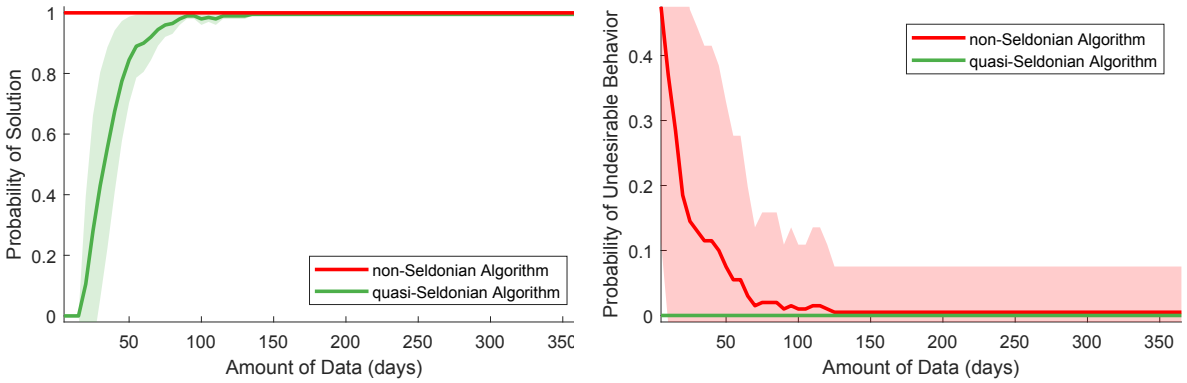


Fig. S24. (Left) The probability that a solution other than No Solution Found (NSF) is returned as the number of days of data is varied. (Right) The probability that a new distribution over policies was returned that increases the prevalence of low blood glucose (defined using r_1) over the initial dosage policy distribution. The shaded region depicts standard deviation over 200 trials.

subset of the reasonable ranges for these parameters ($CR \in [5, 50]$ and $CF \in [1, 31]$), as depicted in Fig. S21. This admissible set of values for CR and CF is a hard constraint, i.e., the specification of a feasible set. We provided our Seldonian algorithm with this feasible set, which limits the solutions that the algorithm can return because it is reasonable in this application to assume that a physician could provide this initial range of reasonable parameters.

This raises the following question: Was the safety of our quasi-Seldonian algorithm due to our providing it with this feasible set? We contend that the answer to this question is "no." The feasible set that we used contains dangerous policies, that is, policies that cause frequent instances of hypoglycemia for adult#003. Consequently, our example is one in which the physician selected a region of policy space (range for CR and CF) that is dangerous and suboptimal. In cases in which the physician correctly identifies optimal settings for CR and CF, there is no need for tuning, and most algorithms would be safe. The crucial setting is that where the initial range contains suboptimal policies and should be adjusted. Given that the initial range of values for CR and CF that we used includes dangerous policies, it is important that an algorithm that automatically adjusts the treatment policy does not deploy a new treatment policy (within the window specified by the physician) that increases the prevalence of low blood glucose. Fig. S24 shows that this could happen: the non-Seldonian algorithm returned solutions that frequently increased the prevalence of low blood glucose *even though it too was restricted to only return solutions within the feasible set specified by the physician.*

Figures S24 and S25 show that the algorithm behaved as desired. However, these results are not those typically reported when evaluating the efficacy of an adaptive bolus calculator. Maahs et al. [196] suggest several performance metrics for evaluating the efficacy of a diabetes management system. Of the many proposed metrics, we select three to present here: percent of time hypoglycemic (blood glucose at or below 70 mg/dL), percent of time hyperglycemic (blood glucose at or above 180 mg/dL), and mean blood glucose over the day. Because these statistics are not what the Seldonian algorithm was tasked with optimizing or constraining,

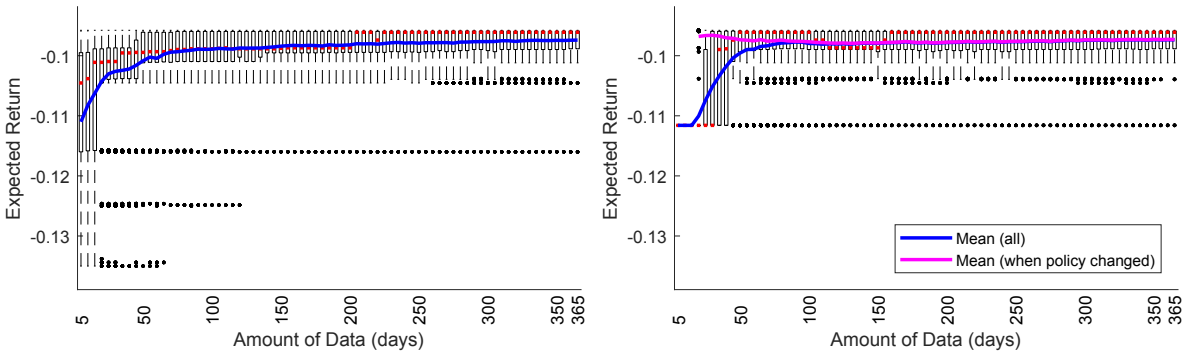


Fig. S25. Left: For different numbers of days of data (in intervals of 5), a box-plot of the distribution of expected returns of the solutions produced by the algorithm designed using the standard ML approach. Outliers are shown as black dots, the blue line is the mean return, and red lines within the boxes mark the medians. All points below -0.1116 (where the plot on the right begins) correspond to cases in which the standard algorithm both decreased performance and produced undesirable behavior (an increase in the prevalence of low blood glucose). **Right:** The same as the left plot, but for the quasi-Seldonian algorithm. The blue line is the mean return, where the initial distribution over policies is used if the algorithm returns NSF, and the magenta line is the mean return given that a solution other than NSF is returned.

reporting these statistics does not quantify the success of the Seldonian algorithm; instead it quantifies the quality of the initial policy distribution, return function, and behavioral constraint that we chose.

Figures S26, S27, and S28 present box plots of the percent time hypoglycemic, percent time hyperglycemic, and mean glucose, respectively. The process for producing these plots was the same as the process described for Fig. S25, except computation of the expected return was replaced with computation of the specified statistic. Fig. S26 shows that the Seldonian algorithm was always effective at reducing the percent of time spent hypoglycemic (when using any amount of data, from 5 days to one year, and across all 200 trials). This is evident by the lack of any boxes denoting outliers above the initial value (which corresponds to the percent of time hypoglycemic per day when using the initial policy distribution). Fig. S27 shows that both algorithms increased the percent of time hyperglycemic per day, as expected, since changes to CR and CF that decrease the time spent hypoglycemic or hyperglycemic cause the other statistic to increase, a property resulting from the combination of this simulated patient, the form of the controller, and the initial ranges for CR and CF. Fig. S28 shows that the Seldonian algorithm produced slightly higher mean glucose.

5.3 Additional Experiments

5.3.1 Different Behavioral Constraints

Fig. S27 shows that our method often increased the percent time spent hyperglycemic. Although this behavior is undesirable, it is not the behavior that we instructed the Seldonian algorithm to avoid. The Seldonian algorithm was instructed to constrain hypoglycemia; not

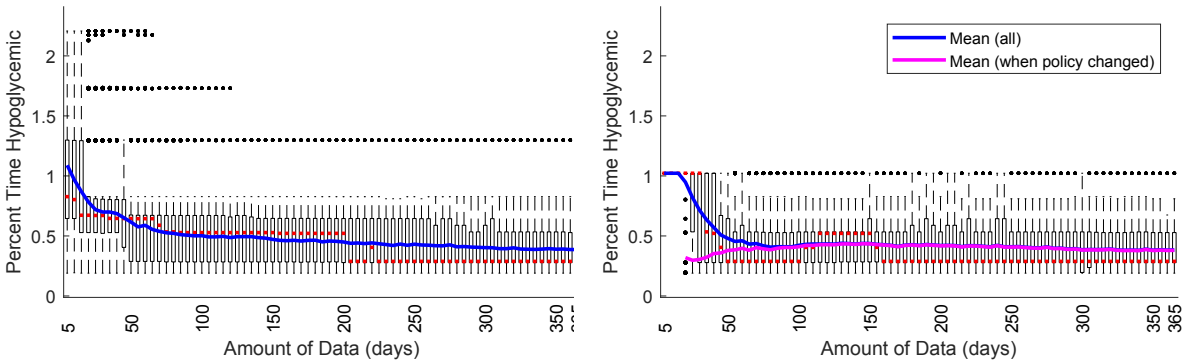


Fig. S26. Box plots showing the percent time hypoglycemic (blood glucose at or below 70 mg/dL) when using the algorithm designed using the standard ML approach (left) and our quasi-Seldonian algorithm (right). The time hypoglycemic per day is measured in days, so that values correspond to the percent of the day spent hypoglycemic. The percent time hypoglycemic of the initial policy distribution is 1.02%.

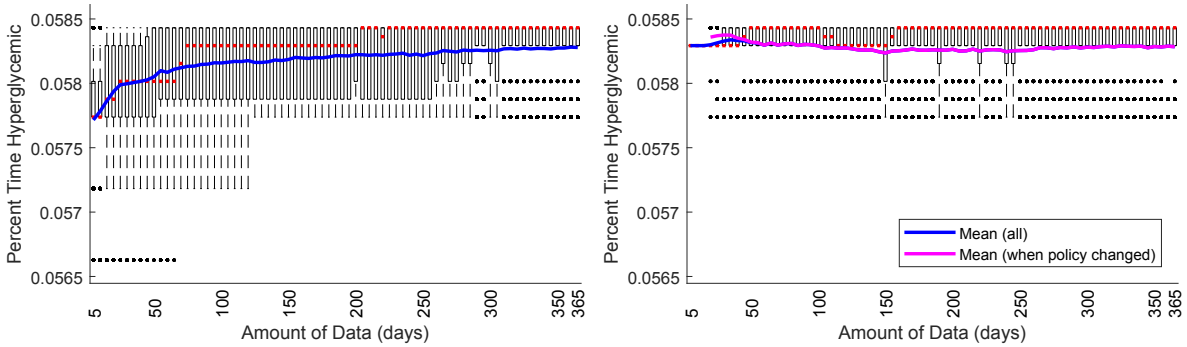


Fig. S27. Box plots showing the percent time hyperglycemic (blood glucose at or above 180 mg/dL) when using the algorithm designed using the standard ML approach (left) and our quasi-Seldonian algorithm (right). The percent time hyperglycemic of the initial policy distribution is 0.0583%.

hyperglycemia. If the user of the Seldonian algorithm had different views of the constraints that should be applied, he or she could apply the algorithm with a different auxiliary return function that better captures his/her desired definitions of undesirable behavior of an adaptive bolus calculator. To show this, we re-ran these same experiments, but using different definitions of $-r_1(h)$ that the user of the algorithm might select, and using fewer trials (32 trials rather than 200).

Specifically, we experimented with three alterantive definitions of $-r_1$. We refer to the original definition in Eq. S39 as the "Hypoglycemic-Return" constraint. The three alternative definitions that we considered are:

- **Time-Hypoglycemic**: the fraction of time per day that the subject was hypoglycemic (blood glucose below 70 mg/DL):

$$r(h) := \frac{1}{1441} \sum_{t=0}^{1440} \begin{cases} -1 & \text{if } g_t \leq 70\,\text{mg/dL} \\ 0 & \text{otherwise.} \end{cases} \tag{S40}$$

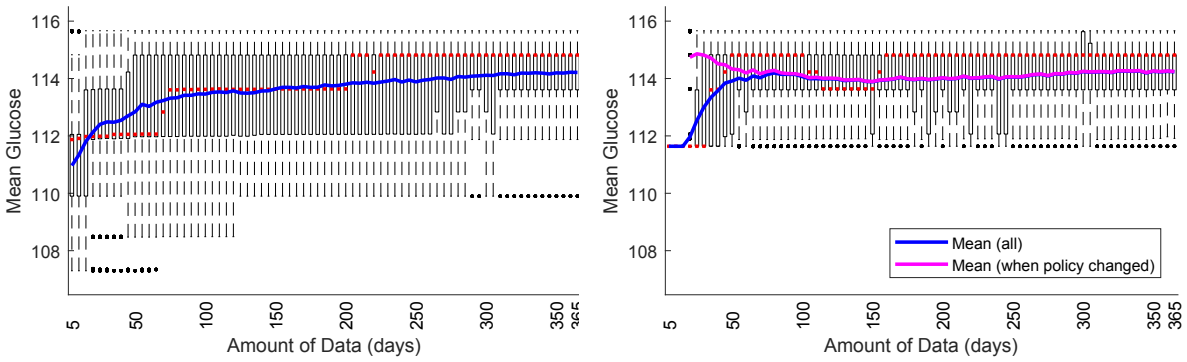


Fig. S28. Box plots showing the mean blood glucose (in mg/dL) of the policies produced by the algorithm designed using the standard ML approach (left) and our quasi-Seldonian algorithm (right). The mean glucuse produced using the initial policy distribution is 111.63 mg/dL.

- **Time-Hyperglycemic**: the fraction of time per day that the subject was hyperglycemic (blood glucose above 180 mg/DL):

$$r(h) \coloneqq \frac{1}{1441} \sum_{t=0}^{1440} \begin{cases} -1 & \text{if } g_t \geq 180\,\text{mg/dL} \\ 0 & \text{otherwise.} \end{cases} \tag{S41}$$

- **Expected Return**: the auxiliary return was defined to be the same as the return defined by Eq. S38. Because the return trades-off hypoglycemia and hyperglycemia, this definition of the auxiliary function allows some increase in hypoglycemia if it results in a sufficiently large decrease in the prevalence of hyperglycemia, and vice versa.

The key messages from these additional experiments are that 1) the algorithm was again able to return solutions using reasonable amounts of data, when safe solutions exist, and 2) when instructed to bound a particular statistic, the Seldonian algorithm did so without fail in all cases.

Consider first the experiment using the time-hypoglycemic constraint. The results from this experiment are reported in Figures S29 and S30. Fig. S29 shows that, although the Seldonian algorithm still returned solutions using a reasonable amount of data, more data was required than when the original definition of r_1 was used. This is likely because the original definition of r_1 produced non-zero values for a wider range of blood glucose levels, thereby providing a less sparse signal. Fig. S29 also shows that the Seldonian algorithm successfully enforced the behavioral constraints, with the probability of undesirable behavior remaining well below the specified 5% threshold.

Next we repeated our experiments using the time-hyperglycemic constraint. The results of these experiments are reported in Figures S31 and S32. Although our original system resulted in changes to the policy distribution that increased the prevalence of hyperglycemia, this example shows the behavior of our algorithm when it is explicitly required (via a behavioral constraint) to ensure that the time spent hyperglycemic is not increased. This is a particularly interesting example because, due to the experimental design (the type of controller, initial ranges for CR and CF, and the chosen simulated subject), policies that increase the expected

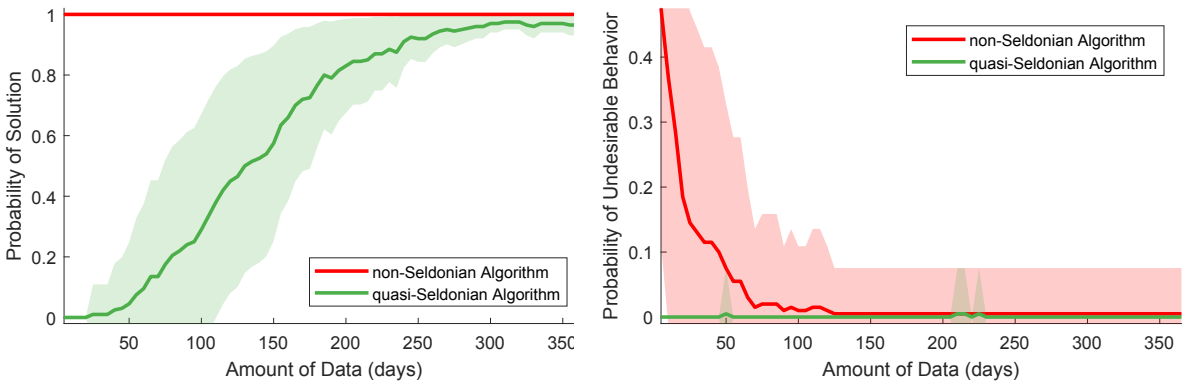


Fig. S29. Results using the time-hypoglycemic constraint. (Left) The probability that a solution other than NO SOLUTION FOUND (NSF) is returned as the number of days of data is varied. (Right) The probability that a distribution over policies was returned that increased the prevalence of undesirable behavior (increased the expected time hypoglycemic) relative to the initial distribution over policies. The shaded region depicts standard deviation, measured over 32 trials.

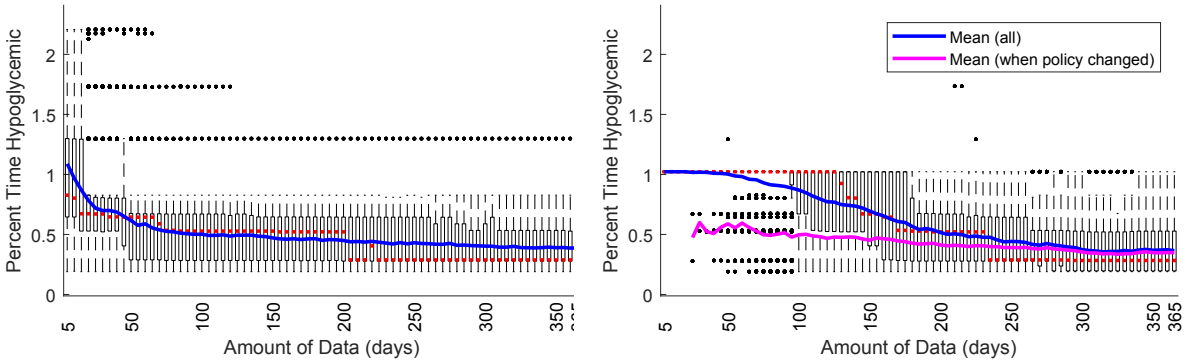


Fig. S30. Results using the time-hypoglycemic constraint. Box plots showing the percent time hypoglycemic (blood glucose at or below 70 mg/dL) when using the algorithm designed using the standard ML approach (right) and our quasi-Seldonian algorithm (left). The percent time hypoglycemic of the initial policy distribution is 1.02%.

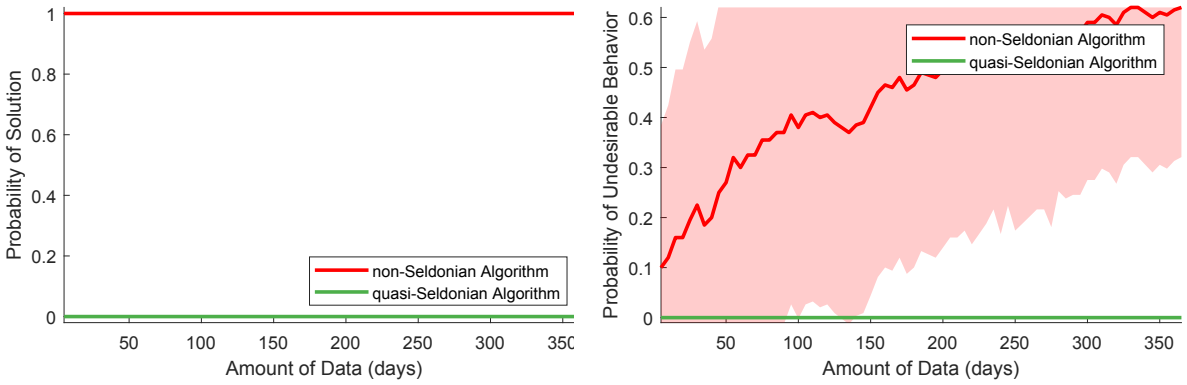


Fig. S31. Results using the time-hyperglycemic constraint. (Left) The probability that a solution other than No Solution Found (NSF) was returned as the number of days of data was varied. (Right) The probability that a distribution over policies was returned that increased the prevalence of undesirable behavior (increased the expected time hyperglycemic) relative to the initial distribution over policies. The shaded region depicts standard deviation, measured over 32 trials.

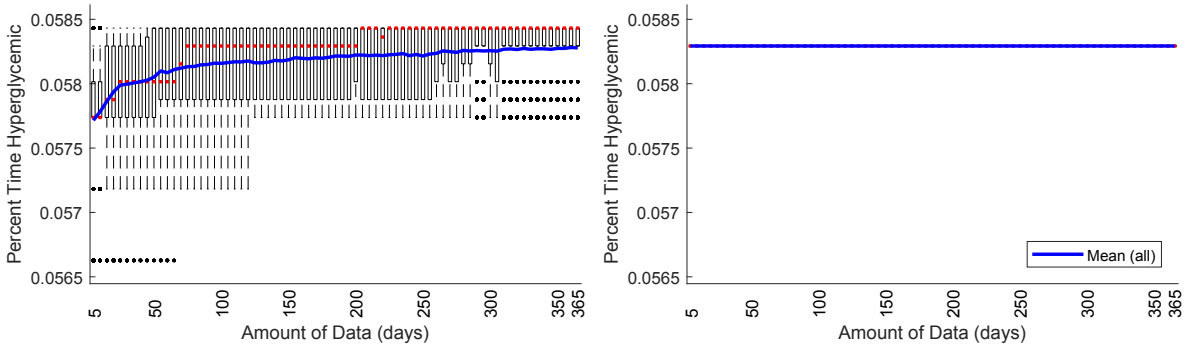


Fig. S32. Results using the time-hyperglycemic constraint. Box plots showing the percent time hyperglycemic (blood glucose at or above 180 mg/dL) when using the algorithm designed using the standard ML approach (left) and our quasi-Seldonian algorithm (right). The standard algorithm still increased the percent time hyperglycemic per day, even when using a year of data, because the return function (primary objective) was not modified—only the behavioral constraint was modified from the original experiment. The percent time hyperglycemic of the initial policy distribution is 0.0583%.

return (the primary objective) all also increase the prevalence of hyperglycemia. This is a result of the fact that the return function penalizes hypoglycemia more than hyperglycemia. For this case, we verified that all 27 possible distributions over policies that the RL agent could return would either violate this safety constraint or decrease the expected return. Our algorithm therefore exhibited ideal behavior in this example: it returned NoSolutionFound on every trial.

Finally, we repeated our experiments using the expected return constraint. The results from this experiment are reported in Figures S33 and S34. While the standard algorithm sometimes returns policies that decrease the expected return, the Seldonian algorithm successfully ensures that (with probability at least 95%) it only changes the policy distribution when the expected return will increase.

5.3.2 Reducing Hypoglycemia and Hyperglycemia Given a Different Initial Policy Distribution

In some scenarios our algorithm can reduce the frequency of both hypoglycemia and hyperglycemia relative to an initial policy distribution: whether this is possible or not depends on the initial policy distribution and the chosen behavioral constraint. To illustrate this, we consider a case where the initial policy distribution is defined over a different and wider range: $CR \in [20, 30]$ and $CF \in [5, 10]$ rather than $CR \in [8.5, 11]$ and $CF \in [10, 15]$. Referring back to Fig. S22, it is clear that this larger window contains worse initial values for CR and CF, which cause both hypoglycemia and hyperglycemia. This example reflects the case where a clinician initially has more uncertainty about which parameter values will work well for a patient, and where improvement with respect to both time hypoglycemic and time hyperglycemic is possible.

We repeated our experiment with adult#003, and the initial definition of $-r_1$ that penalizes hypoglycemia proportional to its severity, but with this modified initial range for CR and CF. The results of this experiment are presented in Fig. S35, which shows that the Seldonian algorithm was able to return solutions using a small amount of data, and that these solutions decreased the percent time hypoglycemic as well as the percent time hyperglycemic. In other words, when a reduction in both hypoglycemia and hyperglycemia is possible, the Seldonian algorithm is capable of reducing both while enforcing a behavioral constraint (here on hypoglycemia).

5.3.3 Different Simulated Subjects

Because our method improves *personalized* treatment, our discussion above focused on running multiple trials using an individual patient. However, Version 3.2 of T1DMS includes ten simulated adult subjects. We repeated our experiments using all ten of these simulated adults. The goal of this experiment was to provide further evidence that our algorithm can return solutions given a reasonable amount of data, and that it properly ensures that the prevalence of undesirable behavior (in whatever manner it is specified) is not increased.

We did not change the initial values for CR and CF. This provides variability across our experiments because these ranges for CR and CF are poor choices for some simulated patients. As shown by the results to be described here, this variability had no impact on our algorithm's behavior, regardless of which simulated patient was used and how "good" the

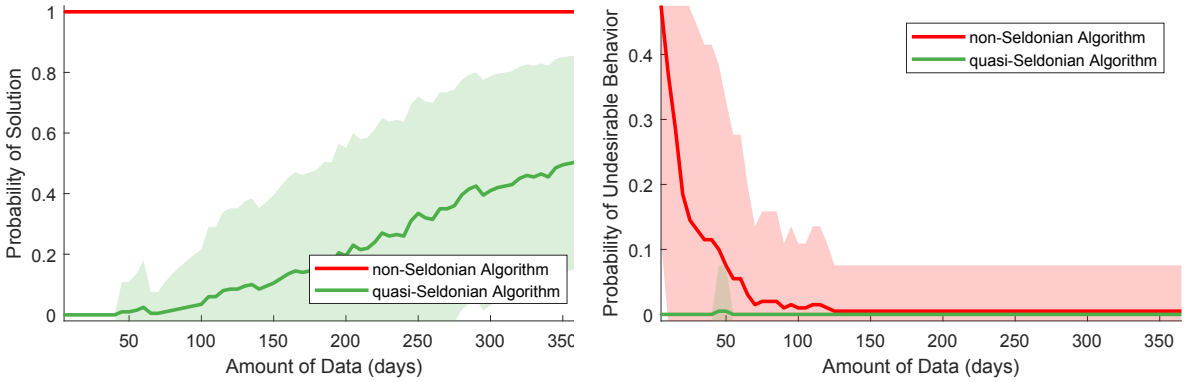


Fig. S33. Results using the expected return constraint. (Left) The probability that a solution other than No Solution Found (NSF) was returned as the number of days of data was varied. (Right) The probability that a distribution over policies was returned that increased the prevalence of undesirable behavior (decreased the expected return) relative to the initial distribution over policies. The shaded region depicts standard deviation, measured over 32 trials.

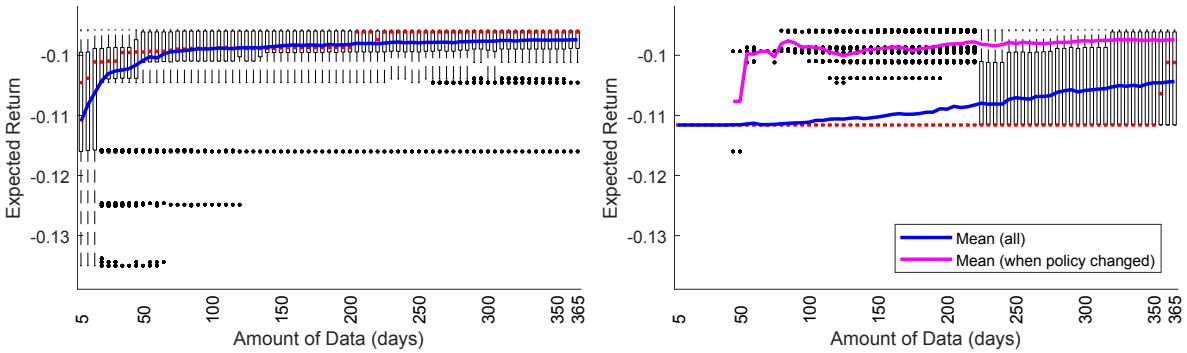


Fig. S34. Results using the expected return constraint. Box plots showing the expected returns that result when using the algorithm designed using the standard ML approach (left) and our quasi-Seldonian algorithm (right). The expected return of the initial policy distribution is -0.1116.

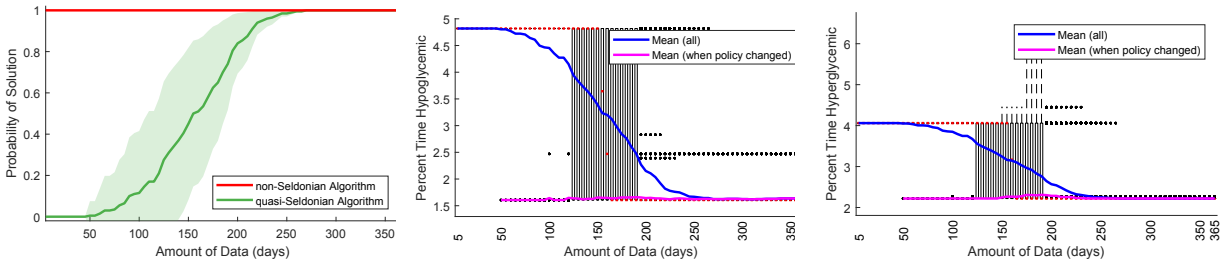


Fig. S35. Results using the different initial range for CR and CF, averaging over 32 trials. The left plot shows the probability a solution other than NSF was returned. We omit the plot showing the probability of undesirable behavior because in this example undesirable behavior never occurred. The middle and right plots show the percent time hypoglycemic and percent time hyperglycemic when using the quasi-Seldonian algorithm (compare to Figures S26 and S27 respectively). This differs from previous plots: here both box plots are for the quasi-Seldonian method, but present different metrics.

initial ranges for CR and CF were, our algorithm reliably found policies that increased the expected return (when such policies existed) while enforcing the chosen behavioral constraints. We note that this variability implies that the magnitudes of the reported statistics, such as time hypoglycemic, time hyperglycemic, and mean glucose, are not meaningful because these statistics depend heavily on the quality of the initial range for CR and CF for each particular patient. However, the relative magnitudes of these statistics before and after the distribution over policies is changed remain meaningful. Our algorithm should increase expected return while ensuring that the behavioral constraints are enforced, even if this means only requiring improvement relative to a poor initial distribution over policies, e.g., one that causes frequent hypoglycemia.

The results of these experiments are summarized in Figures S36 through S39. Each of these figures presents the probability of a solution being returned on the left, and the probability of undesirable behavior being produced on the right. Each curve in each figure panel corresponds to a different simulated adult patient, and all panels show ten curves using both the quasi-Seldonian and non-Seldonian algorithm (although some curves are obscured by other curves). For each possible behavioral constraint and each simulated adult, 32 trials were run and standard deviation error bars are included.

Fig. S36 depicts the results when using the original definition of the auxiliary return function, $-r_1$, which limits the prevalence of low blood glucose by punishing hypoglycemia in proportion to its severity. Fig. S37 depicts the results when using the time-hypoglycemic definition of the auxiliary return function, which limits the fraction of time per day that the person is hypoglycemic while not taking into account the severity of the hypoglycemia beyond the 70 mg/dL threshold. Fig. S38 depicts the results obtained using the time-hyperglycemic definition of the auxiliary return function, which limits the fraction of time per day that the person is hyperglycemic. Fig. S39 depicts the results obtained using the expected return definition of the auxiliary return function, which requires improvement with respect to the primary return function that trades-off hyperglycemia and hypoglycemia.

A salient feature of these results is that regardless of the simulated subject and the chosen definition of the behavioral constraint, the quasi-Seldonian algorithm behaved precisely as

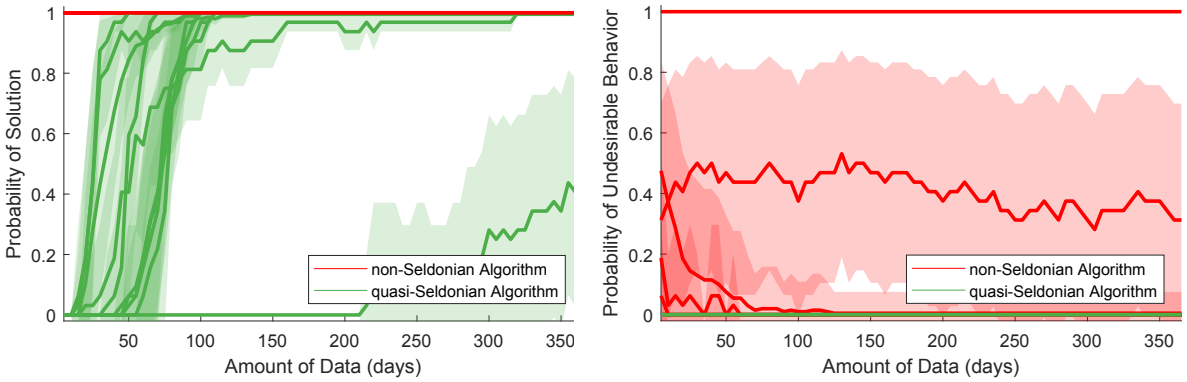


Fig. S36. Results on ten *in silico* subjects using the original definition of r_1.

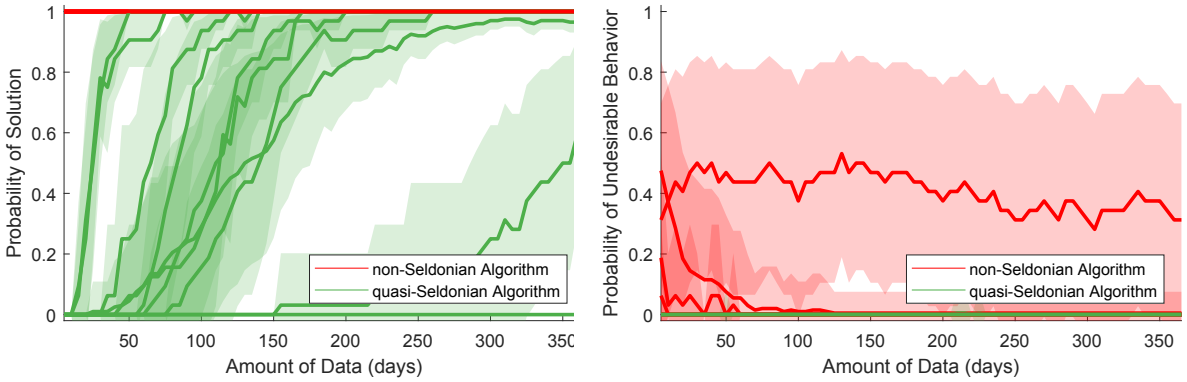


Fig. S37. Results on ten *in silico* subjects using the time-hypoglycemic definition of r_1.

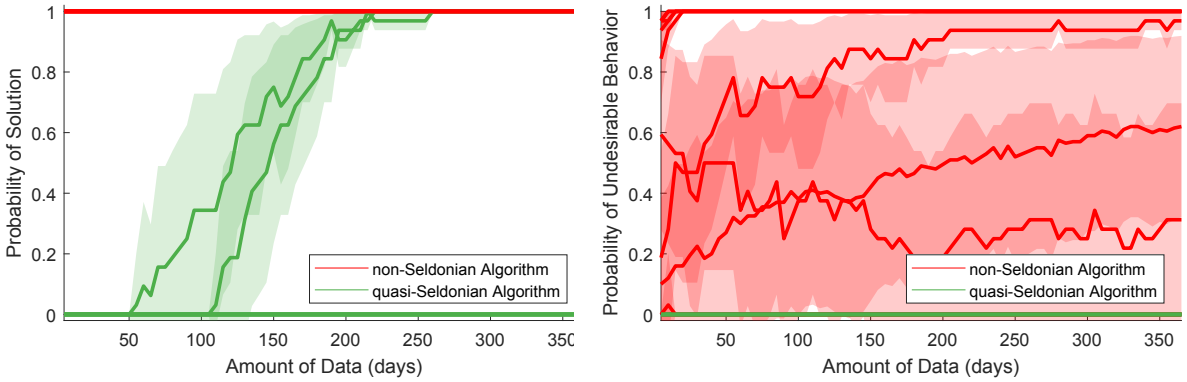


Fig. S38. Results on ten *in silico* subjects using the time-hyperglycemic definition of r_1.

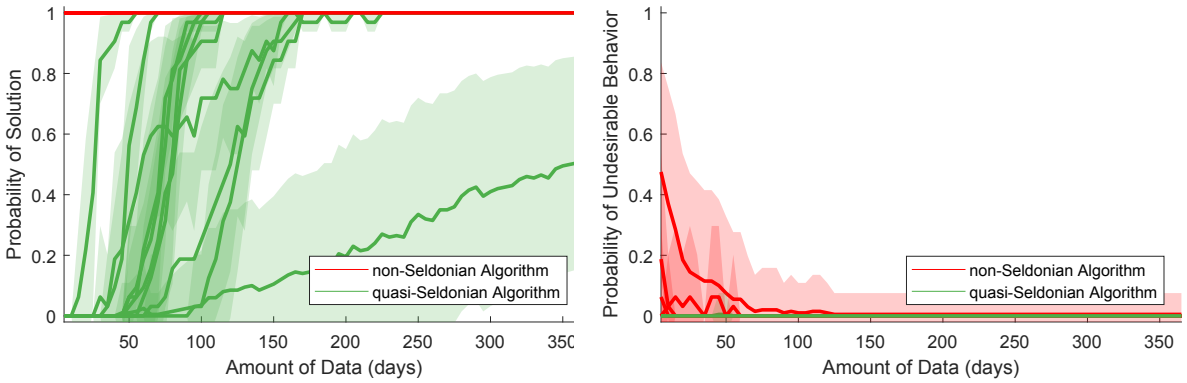


Fig. S39. Results on ten *in silico* subjects using the expected return definition of r_1.

desired. It returned solutions using a reasonable amount of data when possible,[8] and in all cases the probability that the quasi-Seldonian algorithm produced undesirable behavior was well below the required 5% limit. In contrast, the non-Seldonian algorithm frequently produced policy distributions that produced undesirable behavior.

5.4 Additional Considerations for Clinical Applications

New machine learning algorithms are often validated by applying them to mockups of real applications. Our example application of a Seldonian RL algorithm to a mockup of the adaptive bolus calculation problem shows the feasibility of creating and using Seldonian algorithms. This example also suggests that an interdisciplinary team of computer scientists and medical researchers could pursue the actual use of (Seldonian) RL algorithms for adaptive bolus calculation. An actual application of this sort would require several additional considerations, some of which we review here.

First, recall that our experimental design mimics that of prior work applying standard RL algorithms to bolus insulin calculation [30]. An actual deployment of RL to this problem would likely involve a more sophisticated and modern approach. For example, the RL algorithm could treat the target blood glucose as an adaptable parameter of the policy and it could modify the basal rate in conjunction with the bolus dose. Similarly, a more sophisticated policy class (expression for the bolus calculator) would consider insulin on board and estimates of past and anticipated future physical activity [197, 198].

Second, notice that the Seldonian framework is a framework for designing algorithms, not a particular algorithm. As such, there are many possible Seldonian RL algorithms. While the one that we used here provided promising results with our experimental design, a different experimental design might be better paired with a different Seldonian algorithm. For example, if a more sophisticated policy representation is used, e.g., an artificial neural network, then an algorithm like the one in Fig. S15 might be better suited, since it does not assume that there is a small set of possible policies to consider.

Furthermore, in an effort to keep our algorithm simple, we considered distributions over

[8]For all but two simulated subjects, performance with respect to the chosen definition of the primary return function is not possible without increasing the percent time hypoglycemic, and so solutions are typically not returned in Fig. S38.

deterministic policies that can be viewed as rectangles in CR and CF space, as depicted in Figures S21 and S23. The behavior of the resulting controllers is therefore easily interpretable: each day the CR and CF parameters will be sampled uniformly randomly from the specified range. A more conventional RL approach would be to maintain a single stochastic policy, perhaps using the Fourier basis [199] and softmax action selection [47], rather than a distribution over deterministic policies. This would provide a more expressive policy class and would result in a more conventional RL formulation that allows for the use of more advanced off-policy evaluation techniques than the basic importance sampling estimator that we used [149, 195]. However, the solution output by the RL algorithm using this approach would be far less interpretable—a large vector of real-valued numbers. Hence, practitioners must decide between the interpretability of the approach that we have taken and the possibly improved performance of a more conventional (yet still Seldonian) RL approach.

Next, recall that in Eq. S38 we adopted the primary objective function used in prior work [30]. This objective function penalizes deviations of blood glucose from the target blood glucose level, with a non-linear relationship between blood glucose deviation and the resulting penalty. The precise shape of this objective function defines optimal behavior, and therefore has a significant impact on the policies that are returned. Alternatives might vary the rate at which the penalty grows, or might provide no penalty as long as blood glucose is within some range of the target blood glucose.

Like the choice of the primary objective function, practitioners must decide what definition of undesirable behavior is appropriate for this application. Should the algorithm guarantee that the mean time hypoglycemic is not increased? Should the mean time hypoglycemic be weighted by the severity of the hypoglycemia as in what we called the *prevalence of low blood glucose*? Should slight increases in the mean time mildly hypoglycemic be allowed if there is a significant decrease in time hyperglycemic? If so, how should the terms "slight", "mildly", and "significant" be quantified? While Seldonian algorithms provide the user with an interface to define undesirable behavior, it is still up to the practitioner to make these decisions about what consistutes undesirable behavior for their application.

Finally, a clinical application should likely be preceded by further simulation studies that more closely model the clinical application. These simulations should use the policy class, Seldonian algorithm, primary objective function, and definitions of undesirable behavior chosen for the clinical application. Furthermore, these simulations might include additional details not included in our study. For example, the simulation might incorporate physical activity, which can have a significant impact on blood glucose levels in a person with type 1 diabetes [200], or could incorporate more accurate simulations of the delay between when insulin is injected and when the insulin pump measures a change in blood glucose. These additional simulations would provide insight into how the chosen Seldonian algorithm could be expected to perform in the intended clinical application.

The complete enumeration of how each possible change to the experimental design can be expected to influence the performance of a Seldonian RL algorithm is closely tied to the particular algorithm that is chosen, and is beyond the scope of this work. For further information regarding when RL algorithms are and are not effective, we refer the reader to texts on RL [47, 193]. For further information regarding when *Seldonian* RL algorithms are and are not effective, we refer the reader to work regarding challenges with (high-confidence) off-policy evaluation—the component of our Seldonian RL algorithms that is most sensitive

to the experimental design [136, 201, 202, 203].

6 Other Seldonian Algorithms

There has been growing interest in ensuring that machine learning algorithms are safe to use [8, 7, 64]. Given the practical nature of the guarantees provided by Seldonian algorithms, it would be surprising if there were not already some algorithms that can be viewed as (quasi-)Seldonian algorithms for specific Seldonian optimization problems. Here we discuss some existing (quasi-)Seldonian algorithms and the problems that they solve. However, to the best of our knowledge, the framework of Seldonian optimization problems has not been proposed as a general problem formulation for machine learning, nor has its benefits been thoroughly discussed previously.

One example of both Seldonian and quasi-Seldonian algorithms is our prior work to create reinforcement learning algorithms for digital marketing applications [136, 26, 137]. These algorithms observe a vector of features describing the information that is known about a person visiting a webpage, and they decide which advertisement, or which type of advertisement, to display on the webpage. The deployment of a policy that is worse than existing policies would result in fewer clicks on the advertisements. This in turn could be costly both in terms of lost advertisement revenue and lost customers for a digital marketing product.

In the context of digital marketing, we proposed Seldonian and quasi-Seldonian batch reinforcement learning algorithms that guarantee that with probability at least $1 - \delta$, their performance (in terms of the expected return of the MDP) will be at least some constant, ρ_-, where the user of the algorithm is free to select δ and ρ_-. Although this prior work was a steppingstone towards this work, it lacked several important features. First, we did not allow for any other behavioral constraints; the only constraint we considered was to ensure that the expected return was increased with high probability. Second, we only considered the reinforcement learning setting; we did not observe that behavioral constraints could be important for other branches of machine learning.

Others have considered the problem of guaranteeing policy improvement with high confidence [204, 28], sometimes in the context of determining how the complexity of (approximately) solving MDPs grows with different MDP parameters, such as the number of possible states [205]. Also considered by others is how to construct confidence intervals around performance estimates on the basis of *counterfactual reasoning* [206]. This approach is similar to, and precedes, our own prior work [136]. As in our earlier work, these examples address only one special case of SOPs, specifically, SOPs that model reinforcement learning (or bandit) problems and contain the single behavioral constraint that performance (in terms of the standard objective function) should be increased with high probability. This single behavioral constraint is common and not unique to reinforcement learning. The problem of bounding the generalization error of a supervised learning algorithm has been extensively studied [88].

Another example of a (quasi-)Seldonian algorithm is the algorithm presented by Berkenkamp et al. [207], which was developed in parallel with, and independently of, our present approach. Berkenkamp et al. [207] propose a special case of the Seldonian optimization problem framework in which the goal is to tune the hyperparameters of a control algorithm to ensure that,

with high probability, the controller will avoid unsafe regions of state space. These authors propose a quasi-Seldonian algorithm that uses Gaussian processes to approximate both the objective function and behavioral constraint functions, and which then acts conservatively with respect to the confidence bounds produced by the Gaussian process. This example is another instance supporting the utility and practicality of the Seldonian optimization problem formulation, though Berkenkamp et al. [207] do not discuss how their problem framework can be generalized to other applications.

Yet another example of Seldonian algorithms are data-driven robust optimization algorithms [208, 29, 209, 210]. Whereas we present desired properties of a safe algorithm and allow the designer of an ML algorithm freedom with respect to how to satisfy these properties, the data-driven robust optimization framework presents one particular way that these properties could be enforced. So far these algorithms have been restricted to convex constraints and assume that the primary objective has a particular structural form, making them unsuitable for the example applications presented in this work. Furthermore, none of the Seldonian algorithms that we have presented take the particular approach prescribed by the data-driven robust optimization framework. However, data-driven robust optimization algorithms provide powerful and particularly computationally efficient Seldonian algorithms for problems that satisfy the assumptions of these approaches, and an interesting area of future work is to extend such approaches towards weaker assumptions on the problem setting.

In addition, one might view many algorithms as solutions to specific Seldonian optimization problems. For example, many of the state-avoidant algorithms we discussed when describing hard constraints, e.g., the algorithms proposed by Akametalu et al. [63], ensure that with high probability they will not allow a system to enter a pre-specified undesirable state. These algorithms can therefore be viewed as (quasi-)Seldonian algorithms for an SOP that includes the behavioral constraint that, with high probability, the agent will never enter an undesirable state (which requires that at least some prior knowledge about the system dynamics is available). In fact, it is fair to say that since the Seldonian optimization framework subsumes the standard machine learning optimization framework, most existing machine learning algorithms can be viewed as instances of (quasi-)Seldonian algorithms.

7 Future Work

We have presented a framework for designing well-behaved machine learning algorithms, called *Seldonian algorithms*. These Seldonian algorithms provide their users with an interface for defining undesirable behavior in a way that is appropriate for the task at hand, e.g., unsafe behavior, unfair behavior, or unprofitable behavior, and also allow the user to specify δ, the maximum admissible probability of this undesirable behavior. A Seldonian algorithm then guarantees that the probability that it will return a solution that produces undesirable behavior is at most δ. This shifts much of the difficulty of ensuring that a machine learning algorithm is well-behaved from the user of the machine learning algorithm to the designer of the machine learning algorithm, thereby making it easier for the user of a machine learning algorithm to control its behavior.

The purpose of this paper is to introduce the Seldonian framework, provide motivation for their adoption, and present examples showing that the design of practical Seldonian algorithms is tractable. Critically, while the Seldonian algorithms that we have presented

achieve impressive results, they remain simple algorithms that could be improved in many ways. In this section we discuss possible directions of future work, including both algorithmic improvements and possible extensions of our framework.

7.1 Algorithmic Improvements

When designing our example Seldonian algorithms, we encountered many new research questions that might be studied by machine learning researchers. Principled answers to these questions would result in algorithms that return solutions other than NSF given less data, algorithms that better optimize the primary objective function while satisfying the behavioral constraints, faster run times, and improved interfaces for defining undesirable behavior.

For the algorithm in Fig. S15 these open questions include: **1)** can the split of D into D_1 and D_2 be phrased as an optimal stopping problem wherein points are incrementally moved from D_2 to D_1? **2)** Is the doubling of the confidence interval during the computation of θ_c (notice that we double the confidence interval from Student's t-test) sufficient to ensure that, if there exists a solution θ such that $g(\theta) \leq -\epsilon$ for some small positive constant ϵ, then in the limit as the number of points in D goes to infinity, the probability that the algorithm returns NSF goes to zero? **3)** Is the decision to select and test a single candidate solution optimal, or is it better in practice to select and test multiple candidate solutions, perhaps using techniques like the reusable holdout [211]? **4)** Can concentration inequalities that are robust to covariate shift [212] provide guarantees when the distribution of data in the future may differ slightly from the distribution from which the training data was sampled? **5)** Do there exist optimization algorithms that are particularly well-suited to approximating a solution to the constrained global maximization problem that defines θ_c?

Beyond the algorithm in Fig. S15, open questions include: **1)** how can the interface for defining undesirable behaviors be extended to enable users with less experience with machine learning and computers? For example, can the interface be extended to allow undesirable behavior to be defined using natural language? **2)** When is the approach used in our regression and classification algorithms, where one solution is selected and then tested, superior to the approach used in our reinforcement learning algorithm, where all solutions are tested, and then one of the passing solutions selected? **3)** In our Seldonian classification algorithm, where confidence intervals on base variables are propagated through an analytic expression to obtain confidence intervals on $g(\theta)$, can the decisions about when to use one or two-sided confidence intervals on the base variables be optimized using some of the available data? **4)** How can a Seldonian algorithm produce both human-interpretable solutions and a human-interpretable presentation of the evidence that the returned solution is safe? **5)** How can a Seldonian algorithm explain to the user *why* it returned NSF, e.g., do constraints appear to be conflicting, or is there just not enough data?

7.2 Framework Extensions

There are many possible extensions of our framework. For example, it might be extended to replace the single primary objective with multiple primary objectives, resulting in a multi-objective variant, it might be modified to use a Bayesian approach, or it might be extended to use verification techniques [213].

One extension stands out: the extension to the online (sequential) setting, wherein additional data becomes available over time, and the machine learning algorithm is tasked with improving the solution that it returns as more data becomes available. Simply applying a batch Seldonian algorithm (the type we have presented in this paper) multiple times ensures that the probability of a solution that produces undesirable behavior is at most δ *each time the algorithm is run*. Hence, if the algorithm is run k times, the probability that it returns a solution that produces undesirable behavior will be at most $\min\{1, k\delta\}$. An alternative extension to the sequential setting would require the probability of a solution being deployed that produces undesirable behavior to be at most δ, even if the algorithm is run k times, perhaps even when $k \to \infty$. The design of a sequential Seldonian algorithm of this sort may be feasible using *confidence sequences* [214]. Such an extension would also raise new algorithmic questions like, if a sequential Seldonian algorithm resembles our regression and classification algorithms, how should the data within D_1 and D_2 be reused across multiple runs of the algorithm?

Appendix A: Derivation of Minimum MSE Estimator for Illustrative Example

In this appendix we show that the minimum MSE estimator of Y given X in our illustrative example is $\frac{2}{3}X$. Here we do not limit our search of estimators to linear functions—$\frac{2}{3}X$ has the lowest MSE of all possible estimators that use X (and only X) to predict Y. To show this result, we derive an expression for the minimum MSE estimate of Y given that $X = x$.

We begin by writing an expression for the MSE of any estimate, $\hat{y} \in \mathbb{R}$, given that $X = x$.

$$\text{MSE}(\hat{y}) \coloneqq \int_{-\infty}^{\infty} \Pr(Y = y|X = x)(\hat{y} - y)^2 \mathrm{d}y, \tag{S42}$$

where, in this section of the appendix only, we abuse notation and write Pr to denote probability densities rather than probabilities. To find the value of $\hat{y}$ that minimizes MSE we find the critical points of MSE:

$$\begin{aligned} 0 =& \frac{\partial}{\partial \hat{y}} \text{MSE}(\hat{y}) && \text{(S43)} \\ =& \frac{\partial}{\partial \hat{y}} \int_{-\infty}^{\infty} \Pr(Y = y|X = x)(\hat{y} - y)^2 \, \mathrm{d}y && \text{(S44)} \\ =& 2 \int_{-\infty}^{\infty} \Pr(Y = y|X = x)(\hat{y} - y) \, \mathrm{d}y. && \text{(S45)} \end{aligned}$$

By Bayes theorem and marginalizing over T, we have that:

$$\begin{aligned} \Pr(Y = y|X = x) =& \frac{\Pr(X = x|Y = y) \Pr(Y = y)}{\Pr(X = x)} && \text{(S46)} \\ =& \frac{\Pr(X = x|Y = y) \sum_{t=0}^{1} \Pr(T = t) \Pr(Y = y|T = t)}{\Pr(X = x)}. && \text{(S47)} \end{aligned}$$

Thus, continuing from Eq. S45 we have that:

$$0 = 2\int_{-\infty}^{\infty} \frac{\Pr(X=x|Y=y)\sum_{t=0}^{1}\Pr(T=t)\Pr(Y=y|T=t)}{\Pr(X=x)}(\hat{y}-y)\,\mathrm{d}y \tag{S48}$$

$$= \frac{2}{\Pr(X=x)}\int_{-\infty}^{\infty} \underbrace{\frac{1}{2\pi}e^{-\frac{(x-y)^2}{2}}}_{\Pr(X=x|Y=y)} \left(\underbrace{0.5}_{\Pr(T=0)} \underbrace{\frac{1}{2\pi}e^{\frac{-(y-1)^2}{2}}}_{\Pr(Y=y|T=0)} + \underbrace{0.5}_{\Pr(T=1)} \underbrace{\frac{1}{2\pi}e^{\frac{-(y+1)^2}{2}}}_{\Pr(Y=y|T=1)} \right) (\hat{y}-y)\,\mathrm{d}y \tag{S49}$$

$$= \frac{1}{4\pi^2 \Pr(X=x)}\int_{-\infty}^{\infty} e^{-\frac{2(x-y)^2+y^2}{4}}(\hat{y}-y)\,\mathrm{d}y. \tag{S50}$$

Thus,

$$\left(\int_{-\infty}^{\infty} e^{-\frac{2(x-y)^2+y^2}{4}}\,\mathrm{d}y\right)\hat{y} = \int_{-\infty}^{\infty} e^{-\frac{2(x-y)^2+y^2}{4}} y\,\mathrm{d}y \tag{S51}$$

$$\left(2\sqrt{\frac{\pi}{3}}e^{-\frac{x^2}{6}}\right)\hat{y} = \left(\frac{4}{3}\sqrt{\frac{\pi}{3}}e^{\frac{-x^2}{6}}\right)x \tag{S52}$$

$$\hat{y} = \frac{2}{3}x. \tag{S53}$$

It is straightforward to verify that this unique critical point is a global minimum of $\mathrm{MSE}(\hat{y})$ (as opposed to a saddle point or maximum).

References

1. R. W. Jibson, Regression models for estimating coseismic landslide displacement. *Eng. Geol.* **91**, 209–218 (2007). doi:10.1016/j.enggeo.2007.01.013

2. M. Bhasin, G. Raghava, Prediction of CTL epitopes using QM, SVM and ANN techniques. *Vaccine* **22**, 3195–3204 (2004). doi:10.1016/j.vaccine.2004.02.005

3. J. Angwin, J. Larson, S. Mattu, L. Kirchner, Machine bias. *ProPublica*, May 2016; www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

4. D. A. Pomerleau, ALVINN: An autonomous land vehicle in a neural network. *Adv. Neural Inform. Process. Syst.* **1**, 305–313 (1988).

5. S. Saria, A $3 trillion challenge to computational scientists: Transforming healthcare delivery. *IEEE Intell. Syst.* **29**, 82–87 (2014). doi:10.1109/MIS.2014.58

6. N. Bostrom, *Superintelligence: Paths, Dangers, Strategies* (Oxford Univ. Press, 2014).

7. S. Russell, Should we fear supersmart robots? *Sci. Am.* **314**, 58–59 (June 2016).

8. D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, D. Mané, Concrete Problems in AI Safety. arXiv 1606.06565 [cs.AI] (25 July 2016).

9. S. Boyd, L. Vandenberghe, *Convex Optimization* (Cambridge Univ. Press, 2004).

10. D. Bertsimas, G. J. Lauprete, A. Samarov, Shortfall as a risk measure: Properties, optimization and applications. *J. Econ. Dyn. Control* **28**, 1353–1381 (2004). doi:10.1016/S0165-1889(03)00109-X

11. A. Charnes, W. W. Cooper, Chance-constrained programming. *Manage. Sci.* **6**, 73–79 (1959). doi:10.1287/mnsc.6.1.73

12. A. Ben-Tal, L. El Ghaoui, A. Nemirovski, *Robust Optimization* (Princeton Univ. Press, 2009).

13. I. Asimov, *Foundation* (Gnome, 1951).

14. See supplementary materials.

15. O. L. Mangasarian, W. N. Street, W. H. Wolberg, Breast cancer diagnosis and prognosis via linear programming. *Oper. Res.* **43**, 570–577 (1995). doi:10.1287/opre.43.4.570

16. L. Weber, "Your résumé vs. oblivion." *Wall Street Journal* (2012); www.wsj.com/articles/SB10001424052970204624204577178941034941330.

17. L. Li, W. Chu, J. Langford, R. E. Schapire, A contextual-bandit approach to personalized news article recommendation. In *International World Wide Web Conference* (2010), pp. 661–670. doi:10.1145/1772690.1772758

18. R. M. Houtman, C. A. Montgomery, A. R. Gagnon, D. E. Calkin, T. G. Dietterich, S. McGregor, M. Crowley, Allowing a wildfire to burn: Estimating the effect on future fire suppression costs. *Int. J. Wildland Fire* **22**, 871–882 (2013). doi:10.1071/WF12157

19. B. Moore, P. Panousis, V. Kulkarni, L. Pyeatt, A. Doufas, Reinforcement learning for closed-loop propofol anesthesia: A human volunteer study. In *Proceedings of the Twenty-Second Innovative Applications of Artificial Intelligence Conference* (2010), pp. 1807–1813; www.aaai.org/ocs/index.php/IAAI/IAAI10/paper/view/1572/2359.

20. K. Grabczewski, W. Duch, Heterogeneous forests of decision trees. In *International Conference on Artificial Neural Networks* (2002), pp. 504–509. doi:10.1007/3-540-46084-5_82

21. D. Dheeru, E. Karra Taniskidou, UCI Machine Learning Repository (2017); http://archive.ics.uci.edu/ml.

22. K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, D. I. Fotiadis, Machine learning applications in cancer prognosis and prediction. *Comput. Struct. Biotechnol. J.* **13**, 8–17 (2015). doi:10.1016/j.csbj.2014.11.005

23. J. Komiyama, A. Takeda, J. Honda, H. Shimao, *Proc. Mach. Learn. Res.* **80**, 2737–2746 (2018).

24. A. Agarwal, A. Beygelzimer, M. Dudík, J. Langford, H. Wallach, A reductions approach to fair classification. *Proc. Mach. Learn. Res.* **80**, 60–69 (2018).

25. M. B. Zafar, I. Valera, M. G. Rodriguez, K. P. Gummadi, Fairness constraints: Mechanisms for fair classification. *Proc. Mach. Learn. Res.* **54**, 962–970 (2017).

26. P. S. Thomas, G. Theocharous, M. Ghavamzadeh, High confidence policy improvement. *Proc. Mach. Learn. Res.* **37**, 2380–2388 (2015).

27. M. Ghavamzadeh, M. Petrik, Y. Chow, Safe policy improvement by minimizing robust baseline regret. *Adv. Neural Inform. Process. Syst.* **29**, 2298–2306 (2016).

28. R. Laroche, P. Trichelair, R. T. des Combes, Safe policy improvement with baseline bootstrapping. *Proc. Mach. Learn. Res.* **97**, 3652–3661 (2019).

29. D. Bertsimas, V. Gupta, N. Kallus, Data-driven robust optimization. *Math. Program.* **167**, 235–292 (2018). doi:10.1007/s10107-017-1125-8

30. M. Bastani, thesis, University of Alberta (2014).

31. S. Schmidt, K. Nørgaard, Bolus calculators. *J. Diabetes Sci. Technol.* **8**, 1035–1041 (2014). doi:10.1177/1932296814532906

32. C. Dalla Man, F. Micheletto, D. Lv, M. Breton, B. Kovatchev, C. Cobelli, The UVA/Padova type 1 diabetes simulator: New features. *J. Diabetes Sci. Technol.* **8**, 26–34 (2014). doi:10.1177/1932296813514502

33. S. W. Suh, E. T. Gum, A. M. Hamby, P. H. Chan, R. A. Swanson, Hypoglycemic neuronal death is triggered by glucose reperfusion and activation of neuronal NADPH oxidase. *J. Clin. Invest.* **117**, 910–918 (2007). doi:10.1172/JCI30077

34. A. J. Bree, E. C. Puente, D. Daphna-Iken, S. J. Fisher, Diabetes increases brain damage caused by severe hypoglycemia. *Am. J. Physiol. Endocrinol. Metab.* **297**, E194–E201 (2009). doi:10.1152/ajpendo.91041.2008

35. E. C. McNay, V. E. Cotero, Impact of recurrent hypoglycemia on cognitive and brain function. *Physiol. Behav.* **100**, 234–238 (2010). doi:10.1016/j.physbeh.2010.01.004

36. D. Precup, R. S. Sutton, S. Dasgupta, Off-policy temporal-difference learning with function approximation. In *Proceedings of the 18th International Conference on Machine Learning* (2001), pp. 417–424; https://dl.acm.org/citation.cfm?id=655817.

37. H. Zisser, L. Jovanovic, F. Doyle III, P. Ospina, C. Owens, Run-to-run control of meal-related insulin dosing. *Diabetes Technol. Ther.* **7**, 48–57 (2005). doi:10.1089/dia.2005.7.48

38. Data related to this publication are available through Harvard Dataverse. DOI: 10.7910/DVN/O35FW8

39. Source code for all experiments is available through Zenodo. DOI: 10.5281/zenodo.3490615

40. T. M. Mitchell, *Machine Learning* (McGraw-Hill, 1997).

41. D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors. *Nature* **323**, 533–536 (1986). doi:10.1038/323533a0

42. A. Liaw, M. Wiener, Classification and regression by random forest. *R News* **2**, 18–22 (2002).

43. B. E. Boser, I. M. Guyon, V. N. Vapnik, A training algorithm for optimal margin classifiers. In *Annual Workshop on Computational Learning Theory* (1992), pp. 144–152. doi:10.1145/130385.130401

44. L. Breiman, Random forests. *Mach. Learn.* **45**, 5–32 (2001). doi:10.1023/A:1010933404324

45. A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks. *Adv. Neural Inform. Process. Syst.* **25**, 1097–1105 (2012).

46. A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc. B* **39**, 1–38 (1977). doi:10.1111/j.2517-6161.1977.tb01600.x

47. R. S. Sutton, A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, ed. 2, 2018).

48. C. Watkins, thesis, University of Cambridge (1989).

49. I. Asimov, *I, Robot* (Gnome, 1950).

50. C. Dwork, M. Hardt, T. Pitassi, O. Reingold, R. Zemel, Fairness through awareness. In *Innovations in Theoretical Computer Science Conference* (2012), pp. 214–226. doi:10.1145/2090236.2090255

51. T. B. Hashimoto, M. Srivastava, H. Namkoong, P. Liang, Fairness without demographics in repeated loss minimization. *Proc. Mach. Learn. Res.* **80**, 1929–1938 (2018).

52. C. C. Miller, "Can an algorithm hire better than a human?" *New York Times*, June 2015; www.nytimes.com/2015/06/26/upshot/can-an-algorithm-hire-better-than-a-human.html.

53. G. B. Dantzig, A. Orden, P. Wolfe, The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pac. J. Math.* **5**, 183–195 (1955). doi:10.2140/pjm.1955.5.183

54. P. S. Thomas, W. Dabney, S. Mahadevan, S. Giguere, Projected natural actor-critic. *Adv. Neural Inform. Process. Syst.* **26**, 2337–2345 (2013).

55. H. Le, C. Voloshin, Y. Yue, Batch policy learning under constraints. *Proc. Mach. Learn. Res.* **97**, 3703–3712 (2019).

56. A. J. Irani, thesis, Georgia Institute of Technology (2015).

57. C. J. Tomlin, thesis, University of California, Berkeley (1998).

58. M. Oishi, C. J. Tomlin, V. Gopal, D. Godbole, Addressing multiobjective control: Safety and performance through constrained optimization. In *International Workshop on Hybrid Systems: Computation and Control* (2001), pp. 459–472. doi:10.1007/3-540-45351-2_37

59. T. J. Perkins, A. G. Barto, Lyapunov design for safe reinforcement learning. *J. Mach. Learn. Res.* **3**, 803–832 (2003).

60. I. M. Mitchell, A. M. Bayen, C. J. Tomlin, A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Trans. Automat. Contr.* **50**, 947–957 (2005). doi:10.1109/TAC.2005.851439

61. A. Hans, D. Schneegaß, A. M. Schäfer, S. Udluft, Safe exploration for reinforcement learning. In *European Symposium on Artificial Neural Networks* (2008), pp. 143–148; https://pdfs.semanticscholar.org/5ee2/7e9db2ae248d1254107852311117c4cda1c9.pdf.

62. E. Arvelo, N. C. Martins, Control Design for Markov Chains under Safety Constraints: A Convex Approach. arXiv 1209.2883 [cs.SY] (8 November 2012).

63. A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, C. J. Tomlin, Reachability-based safe learning with Gaussian processes. In *IEEE Conference on Decision and Control* (2014), pp. 1424–1431. doi:10.1109/CDC.2014.7039601

64. S. Zilberstein, Building strong semi-autonomous systems. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence* (2015), pp. 4088–4092; www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9920/9686.

65. J. Nocedal, S. Wright, *Numerical Optimization* (Springer, ed. 2, 2006).

66. A. Messac, A. Ismail-Yahaya, C. A. Mattson, The normalized normal constraint method for generating the Pareto frontier. *Struct. Multidiscipl. Optim.* **25**, 86–98 (2003). doi:10.1007/s00158-002-0276-1

67. G. F. Smits, M. Kotanchek, Pareto-front exploitation in symbolic regression. *Genet. Program. Theory Pract.* **II**, 283–299 (2005). doi:10.1007/0-387-23254-0_17

68. M. Pirotta, S. Parisi, M. Restelli, Multi-objective reinforcement learning with continuous Pareto frontier approximation. In *Conference on Artificial Intelligence* (2015), pp. 2928–2934; www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9798/9962.

69. R. G. McCoy, H. K. Van Houten, J. Y. Ziegenfuss, N. D. Shah, R. A. Wermers, S. A. Smith, Increased mortality of patients with diabetes reporting severe hypoglycemia. *Diabetes Care* **35**, 1897–1901 (2012). doi:10.2337/dc11-2054

70. L. B. Miller, H. Wagner, Chance-constrained programming with joint constraints. *Oper. Res.* **13**, 930–945 (1965). doi:10.1287/opre.13.6.930

71. A. Prékopa, On probabilistic constrained programming. In *Princeton Symposium on Mathematical Programming* (1970), pp. 113–138. doi:10.1515/9781400869930-009

72. D. Dentcheva, A. Prékopa, A. Ruszczynski, Concavity and efficient points of discrete distributions in probabilistic programming. *Math. Program.* **89**, 55–77 (2000). doi:10.1007/PL00011393

73. A. Nemirovski, On safe tractable approximations of chance constraints. *Eur. J. Oper. Res.* **219**, 707–718 (2012). doi:10.1016/j.ejor.2011.11.006

74. H. Xu, S. Mannor, Probabilistic goal Markov decision processes. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence* (2011), pp. 2046–2052. doi:10.5591/978-1-57735-516-8/IJCAI11-341

75. M. H. Houck, A chance constrained optimization model for reservoir design and operation. *Water Resour. Res.* **15**, 1011–1016 (1979). doi:10.1029/WR015i005p01011

76. I. Gurvich, J. Luedtke, T. Tezcan, Staffing call centers with uncertain demand forecasts: A chance-constrained optimization approach. *Manage. Sci.* **56**, 1093–1115 (2010). doi:10.1287/mnsc.1100.1173

77. Q. Wang, Y. Guan, J. Wang, A chance-constrained two-stage stochastic program for unit commitment with uncertain wind power output. *IEEE Trans. Power Syst.* **27**, 206–215 (2012). doi:10.1109/TPWRS.2011.2159522

78. E. Erdoğan, G. Iyengar, Ambiguous chance constrained problems and robust optimization. *Math. Program.* **107**, 37–61 (2006). doi:10.1007/s10107-005-0678-0

79. D. P. de Farias, B. Van Roy, On constraint sampling in the linear programming approach to approximate dynamic programming. *Math. Oper. Res.* **29**, 462–478 (2004). doi:10.1287/moor.1040.0094

80. G. Calafiore, M. C. Campi, Uncertain convex programs: Randomized solutions and confidence levels. *Math. Program.* **102**, 25–46 (2005). doi:10.1007/s10107-003-0499-y

81. A. Nemirovski, A. Shapiro, Convex approximations of chance constrained programs. *SIAM J. Optim.* **17**, 969–996 (2006). doi:10.1137/050622328

82. J. R. Birge, F. Louveaux, *Introduction to Stochastic Programming* (Springer, 2011).

83. F. Provost, T. Fawcett, Robust classification for imprecise environments. *Mach. Learn.* **42**, 203–231 (2001). doi:10.1023/A:1007601015854

84. J. García, F. Fernández, A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.* **16**, 1437–1480 (2015).

85. S. Kuindersma, R. Grupen, A. G. Barto, Variational Bayesian optimization for runtime risk-sensitive control. In *Robotics: Science and Systems VIII* (2012), pp. 201–206. doi:10.15607/rss.2012.viii.026

86. A. Tamar, Y. Glassner, S. Mannor, Optimizing the CVaR via sampling. In *Conference on Artificial Intelligence* (2015), pp. 2993–2999; www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9429/9972.

87. Y. Chow, M. Ghavamzadeh, Algorithms for CVaR optimization in MDPs. *Adv. Neural Inform. Process. Syst.* **27**, 3509–3517 (2014).

88. Y. S. Abu-Mostafa, M. Magdon-Ismail, H. T. Lin, *Learning from Data: A Short Course* (AMLBook, 2012).

89. P. Massart, *Concentration Inequalities and Model Selection* (Springer, 2007).

90. W. Hoeffding, Probability inequalities for sums of bounded random variables. *J. Am. Stat. Assoc.* **58**, 13–30 (1963). doi:10.1080/01621459.1963.10500830

91. B. Efron, Better bootstrap confidence intervals. *J. Am. Stat. Assoc.* **82**, 171–185 (1987). doi:10.1080/01621459.1987.10478410

92. L. E. Chambless, A. R. Folsom, A. R. Sharrett, P. Sorlie, D. Couper, M. Szklo, F. J. Nieto, Coronary heart disease risk prediction in the atherosclerosis risk in communities (ARIC) study. *J. Clin. Epidemiol.* **56**, 880–890 (2003). doi:10.1016/S0895-4356(03)00055-6

93. A. R. Folsom, L. E. Chambless, B. B. Duncan, A. C. Gilbert, J. S. Pankow, Prediction of coronary heart disease in middle-aged adults with diabetes. *Diabetes Care* **26**, 2777–2784 (2003). doi:10.2337/diacare.26.10.2777

94. M. Petrik, Y. Chow, M. Ghavamzadeh, Safe policy improvement by minimizing robust baseline regret. *Adv. Neural Inform. Process. Syst.* **29**, 2298–2306 (2016).

95. F. Kamiran, T. Calders, Classifying without discriminating. In *International Conference on Computer, Control and Communication* (2009), pp. 1–6. doi:10.1109/IC4.2009.4909197

96. T. Calders, S. Verwer, Three naive Bayes approaches for discrimination-free classification. *Data Min. Knowl. Discov.* **21**, 277–292 (2010). doi:10.1007/s10618-010-0190-x

97. B. T. Luong, S. Ruggieri, F. Turini, k-NN as an implementation of situation testing for discrimination discovery and prevention. In *ACM Conference on Knowledge Discovery and Data Mining* (2011), pp. 502–510. doi:10.1145/2020408.2020488

98. T. Kamishima, S. Akaho, J. Sakuma, Fairness-aware learning through regularization approach. In *International Conference on Data Mining Workshops* (2011), pp. 643–650. doi:10.1109/icdmw.2011.83

99. M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, S. Venkatasubramanian, Certifying and removing disparate impact. In *ACM Conference on Knowledge Discovery and Data Mining* (2015), pp. 259–268. doi:10.1145/2783258.2783311

100. B. Fish, J. Kun, Á. D. Lelkes, A confidence-based approach for balancing fairness and accuracy. In *SIAM International Conference on Data Mining* (2016), pp. 144–152. doi:10.1137/1.9781611974348.17

101. M. Joseph, M. Kearns, J. Morgenstern, A. Roth, Fairness in learning: Classic and contextual bandits. *Adv. Neural Inform. Process. Syst.* **29**, 325–333 (2016).

102. M. Rabin, Incorporating fairness into game theory and economics. *Am. Econ. Rev.* **83**, 1281–1302 (1993).

103. E. Fehr, K. M. Schmidt, A theory of fairness, competition, and cooperation. *Q. J. Econ.* **114**, 817–868 (1999). doi:10.1162/003355399556151

104. A. Falk, U. Fischbacher, A theory of reciprocity. *Games Econ. Behav.* **54**, 293–315 (2006). doi:10.1016/j.geb.2005.03.001

105. A. Datta, S. Sen, Y. Zick, Algorithmic transparency via quantitative input influence. In *IEEE Symposium on Security and Privacy* (2016), pp. 598–617. doi:10.1109/sp.2016.42

106. P. Adler, C. Falk, S. A. Friedler, G. Rybeck, C. Scheidegger, B. Smith, S. Venkatasubramanian, Auditing black-box models by obscuring features. In *IEEE International Conference on Data Mining* (2016), pp. 1–10. doi:10.1109/icdm.2016.0011

107. A. Datta, M. C. Tschantz, A. Datta, Automated experiments on ad privacy settings. In *Proceedings on Privacy Enhancing Technologies* (2015), pp. 92–112. doi:10.1515/popets-2015-0007

108. S. Galhotra, Y. Brun, A. Meliou, Fairness testing: Testing software for discrimination. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering* (2017), pp. 498–510. doi:10.1145/3106237.3106277

109. A. Narayanan, “21 fairness definitions and their politics” (tutorial at the ACM Conference on Fairness, Accountability, and Transparency, 2018); https://fatconference.org/static/tutorials/narayanan-21defs18.pdf.

110. J. M. Kleinberg, S. Mullainathan, M. Raghavan, Inherent trade-offs in the fair determination of risk scores. In *Innovations in Theoretical Computer Science Conference* (2017), pp. 43:1–43:23. doi:10.4230/LIPIcs.ITCS.2017.43

111. S. A. Friedler, C. Scheidegger, S. Venkatasubramanian, On the (im)possibility of fairness. arXiv 1609.07236 [cs.CY] (23 September 2016).

112. P. T. Kim, Data-driven discrimination at work. *William Mary Law Rev.* **58**, 857 (2016).

113. L. Sweeney, Discrimination in online ad delivery. *Commun. ACM* **56**, 44–54 (2013). doi:10.1145/2447976.2447990

114. D. Ingold, S. Soper, “Amazon doesn’t consider the race of its customers. Should it?” *Bloomberg* (21 April 2016); www.bloomberg.com/graphics/2016-amazon-same-day.

115. *Griggs v. Duke Power Co.*, 401 U.S. 424 (1971).

116. A. Chouldechova, Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data* **5**, 153–163 (2017). doi:10.1089/big.2016.0047

117. L. T. Liu, S. Dean, E. Rolf, M. Simchowitz, M. Hardt, Delayed impact of fair machine learning. *Proc. Mach. Learn. Res.* **80**, 3150–3158 (2018).

118. S. Corbett-Davies, E. Pierson, A. Feller, S. Goel, A. Huq, Algorithmic decision making and the cost of fairness. In *ACM Conference on Knowledge Discovery and Data Mining* (2017), pp. 797–806. doi:10.1145/3097983.3098095

119. M. Hardt, E. Price, N. Srebro, Equality of opportunity in supervised learning. *Adv. Neural Inform. Process. Syst.* **29**, 3323–3331 (2016).

120. R. Berk, H. Heidari, S. Jabbari, M. Kearns, A. Roth, Fairness in criminal justice risk assessments: The state of the art. *Sociol. Methods Res.* 10.1177/0049124118782533 (2018). doi:10.1177/0049124118782533

121. M. J. Kusner, J. R. Loftus, C. Russell, R. Silva, Counterfactual fairness. *Adv. Neural Inform. Process. Syst.* **30**, 4066–4076 (2017).

122. G. N. Rothblum, G. Yona, Probably approximately metric-fair learning. *Proc. Mach. Learn. Res.* **80**, 5680–5688 (2018).

123. F. Kamiran, T. Calders, M. Pechenizkiy, Discrimination aware decision tree learning. In *International Conference on Data Mining* (2010), pp. 869–874. doi:10.1109/icdm.2010.50

124. I. Žliobaite, F. Kamiran, T. Calders, Handling conditional discrimination. In *International Conference on Data Mining* (2011), pp. 992–1001. doi:10.1109/icdm.2011.72

125. T. Calders, F. Kamiran, M. Pechenizkiy, Building classifiers with independency constraints. In *International Conference on Data Mining Workshops* (2009), pp. 13–18. doi:10.1109/icdmw.2009.83

126. C. Dwork, N. Immorlica, A. T. Kalai, M. Leiserson, Decoupled classifiers for group-fair and efficient machine learning. *Proc. Mach. Learn. Res.* **81**, 119–133 (2018).

127. S. Yao, B. Huang, New fairness metrics for recommendation that embrace differences. In *Workshop on Fairness, Accountability, and Transparency in Machine Learning* (2017); https://arxiv.org/pdf/1706.09838.pdf.

128. M. Kay, C. Matuszek, S. A. Munson, Unequal representation and gender stereotypes in image search results for occupations. In *Annual ACM Conference on Human Factors in Computing Systems* (2015), pp. 3819–3828. doi:10.1145/2702123.2702520

129. H. Demuth, M. Beale, Neural network toolbox for use with Matlab, Version 4 (2004); http://cda.psych.uiuc.edu/matlab_pdf/nnet.pdf.

130. N. Hansen, The CMA evolution strategy: A comparing review. In *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*, J. Lozano, P. Larrañaga, I. Inza, E. Bengoetxea, Eds. (Springer, 2006), pp. 75–102. doi:10.1007/11007937_4

131. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).

132. A. Maurer, M. Pontil, Empirical Bernstein bounds and sample variance penalization. In *Annual Conference on Learning Theory* (2009), pp. 115–124; www.cs.mcgill.ca/~colt2009/papers/012.pdf#page=1.

133. G. Tesauro, Temporal difference learning and TD-gammon. *Commun. ACM* **38**, 58–68 (1995). doi:10.1145/203330.203343

134. A. Ng, J. Kim, M. Jordan, S. Sastry, Autonomous helicopter flight via reinforcement learning. *Adv. Neural Inform. Process. Syst.* **17**, 799–806 (2004).

135. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015). doi:10.1038/nature14236

136. P. S. Thomas, G. Theocharous, M. Ghavamzadeh, High confidence off-policy evaluation. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence* (2015), pp. 3000–3006; www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/10042/9973.

137. P. S. Thomas, thesis, University of Massachusetts, Amherst (2015).

138. J. Kober, J. Peters, Learning motor primitives for robotics. In *IEEE International Conference on Robotics and Automation* (2009), pp. 2112–2118. doi:10.1109/robot.2009.5152577

139. F. Sehnke, C. Osendorfer, T. Ruckstiess, A. Graves, J. Peters, J. Schmidhuber, Parameter-exploring policy gradients. *Neural Netw.* **23**, 551–559 (2010). doi:10.1016/j.neunet.2009.12.004

140. E. A. Theodorou, J. Buchli, S. Schaal, A generalized path integral control approach to reinforcement learning. *J. Mach. Learn. Res.* **11**, 3137–3181 (2010).

141. F. Stulp, O. Sigaud, http://hal.archives-ouvertes.fr/hal-00738463 (2012).

142. H. Kahn, A. W. Marshall, Methods of reducing sample size in Monte Carlo computations. *J. Oper. Res. Soc. Am.* **1**, 263–278 (1953). doi:10.1287/opre.1.5.263

143. D. Precup, R. S. Sutton, S. Singh, Eligibility traces for off-policy policy evaluation. In *Proceedings of the 17th International Conference on Machine Learning* (2000), pp. 759–766; https://scholarworks.umass.edu/cgi/viewcontent.cgi?article=1079&context=cs_faculty_pubs.

144. D. P. Bertsekas, J. N. Tsitsiklis, *Neuro-Dynamic Programming* (Athena Scientific, Belmont, MA, 1996).

145. G. Theocharous, P. S. Thomas, M. Ghavamzadeh, Personalized ad recommendation systems for life-time value optimization with guarantees. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence* (2015), pp. 1806–1812. doi:10.1145/2740908.2741998

146. P. S. Thomas, E. Brunskill, Importance sampling with unequal support. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence* (2017), pp. 2646–2652; www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14957/14457.

147. R. G. Miller, *Simultaneous Statistical Inference* (Springer, 2012).

148. H. Kahn, "Use of different Monte Carlo sampling techniques" (Tech. Rep. P-766, RAND Corporation, September 1955).

149. N. Jiang, L. Li, Doubly robust off-policy evaluation for reinforcement learning. *Proc. Mach. Learn. Res.* **48**, 652–661 (2016).

150. J. A. Cruz, D. S. Wishart, Applications of machine learning in cancer prediction and prognosis. *Cancer Inform.* **2**, 59–77 (2006). doi:10.1177/117693510600200030

151. M. A. Shipp, K. N. Ross, P. Tamayo, A. P. Weng, J. L. Kutok, R. C. T. Aguiar, M. Gaasenbeek, M. Angelo, M. Reich, G. S. Pinkus, T. S. Ray, M. A. Koval, K. W. Last, A.

Norton, T. A. Lister, J. Mesirov, D. S. Neuberg, E. S. Lander, J. C. Aster, T. R. Golub, Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nat. Med.* **8**, 68–74 (2002). doi:10.1038/nm0102-68

152. Q.-H. Ye, L.-X. Qin, M. Forgues, P. He, J. W. Kim, A. C. Peng, R. Simon, Y. Li, A. I. Robles, Y. Chen, Z.-C. Ma, Z.-Q. Wu, S.-L. Ye, Y.-K. Liu, Z.-Y. Tang, X. W. Wang, Predicting hepatitis B virus–positive metastatic hepatocellular carcinomas using gene expression profiling and supervised machine learning. *Nat. Med.* **9**, 416–423 (2003). doi:10.1038/nm843

153. World Health Organization, *Global Report on Diabetes* (2016); http://apps.who.int/iris/bitstream/10665/204871/1/9789241565257_eng.pdf.

154. C. Toffanin, M. Messori, F. Di Palma, G. De Nicolao, C. Cobelli, L. Magni, *Artificial Pancreas: Model Predictive Control Design from Clinical Experience* (Sage, 2013).

155. R. Hovorka, V. Canonico, L. J. Chassin, U. Haueter, M. Massi-Benedetti, M. O. Federici, T. R. Pieber, H. C. Schaller, L. Schaupp, T. Vering, M. E. Wilinska, Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes. *Physiol. Meas.* **25**, 905–920 (2004). doi:10.1088/0967-3334/25/4/010

156. S. M. Lynch, B. W. Bequette, Model predictive control of blood glucose in type I diabetics using subcutaneous glucose measurements. In *American Control Conference* (2002), pp. 4039–4043. doi:10.1109/acc.2002.1024561

157. R. S. Parker, F. J. Doyle, N. A. Peppas, A model-based algorithm for blood glucose control in type I diabetic patients. *IEEE Trans. Biomed. Eng.* **46**, 148–157 (1999). doi:10.1109/10.740877

158. H. Schaller, L. Schaupp, M. Bodenlenz, M. Wilinska, L. Chassin, P. Wach, T. Vering, R. Hovorka, T. Pieber, On-line adaptive algorithm with glucose prediction capacity for subcutaneous closed loop control of glucose: Evaluation under fasting conditions in patients with type 1 diabetes. *Diabet. Med.* **23**, 90–93 (2006). doi:10.1111/j.1464-5491.2006.01695.x

159. Y. Matsuo, S. Shimoda, M. Sakakida, K. Nishida, T. Sekigami, S. Ichimori, K. Ichinose, M. Shichiri, E. Araki, Strict glycemic control in diabetic dogs with closed-loop intraperitoneal insulin infusion algorithm designed for an artificial endocrine pancreas. *J. Artif. Organs* **6**, 55–63 (2003). doi:10.1007/s100470300009

160. T. Sekigami, S. Shimoda, K. Nishida, Y. Matsuo, S. Ichimori, K. Ichinose, M. Shichiri, M. Sakakida, E. Araki, Comparison between closed-loop portal and peripheral venous insulin delivery systems for an artificial endocrine pancreas. *J. Artif. Organs* **7**, 91–100 (2004). doi:10.1007/s10047-004-0251-2

161. S. Shimoda, K. Nishida, M. Sakakida, Y. Konno, K. Ichinose, M. Uehara, T. Nowak, M. Shichiri, Closed-loop subcutaneous insulin infusion algorithm with a short-acting insulin

analog for long-term clinical application of a wearable artificial endocrine pancreas. *Front. Med. Biol. Eng.* **8**, 197–211 (1997).

162. G. Marchetti, M. Barolo, L. Jovanovic, H. Zisser, D. E. Seborg, An improved PID switching control strategy for type 1 diabetes. *IEEE Trans. Biomed. Eng.* **55**, 857–865 (2008). doi:10.1109/TBME.2008.915665

163. A. E. Panteleon, M. Loutseiko, G. M. Steil, K. Rebrin, Evaluation of the effect of gain on the meal response of an automated closed-loop insulin delivery system. *Diabetes* **55**, 1995–2000 (2006). doi:10.2337/db05-1346

164. G. Steil, A. Panteleon, K. Rebrin, Closed-loop insulin delivery—The path to physiological glucose control. *Adv. Drug Deliv. Rev.* **56**, 125–144 (2004). doi:10.1016/j.addr.2003.08.011

165. S. Soylu, K. Danisman, I. E. Sacu, M. Alci, Closed-loop control of blood glucose level in type-1 diabetics: A simulation study. In *Electrical and Electronics Engineering* (2013), pp. 371–375. doi:10.1109/eleco.2013.6713864

166. K. Amrein, M. Ellmerer, R. Hovorka, N. Kachel, H. Fries, D. Von Lewinski, K. Smolle, T. R. Pieber, J. Plank, Efficacy and safety of glucose control with space GlucoseControl in the medical intensive care unit—an open clinical investigation. *Diabetes Technol. Ther.* **14**, 690–695 (2012). doi:10.1089/dia.2012.0021

167. R. Hovorka, Closed-loop insulin delivery: From bench to clinical practice. *Nat. Rev. Endocrinol.* **7**, 385–395 (2011). doi:10.1038/nrendo.2011.32

168. J. R. Castle, J. H. DeVries, B. Kovatchev, Future of automated insulin delivery systems. *Diabetes Technol. Ther.* **19**, S-67–S-72 (2017). doi:10.1089/dia.2017.0012

169. H. Zisser, L. Robinson, W. Bevier, E. Dassau, C. Ellingsen, F. J. Doyle III, L. Jovanovič, Bolus calculator: A review of four "smart" insulin pumps. *Diabetes Technol. Ther.* **10**, 441–444 (2008). doi:10.1089/dia.2007.0284

170. R. Gondhalekar, E. Dassau, F. J. Doyle III, Periodic zone-MPC with asymmetric costs for outpatient-ready safety of an artificial pancreas to treat type 1 diabetes. *Automatica* **71**, 237–246 (2016). doi:10.1016/j.automatica.2016.04.015

171. B. Kovatchev, D. M. Raimondo, M. Breton, S. Patek, C. Cobelli, In silico testing and in vivo experiments with closed-loop control of blood glucose in diabetes. *IFAC Proc. Vol.* **41**, 4234–4239 (2008). doi:10.3182/20080706-5-KR-1001.00712

172. F. H. El-Khatib, C. Balliro, M. A. Hillard, K. L. Magyar, L. Ekhlaspour, M. Sinha, D. Mondesir, A. Esmaeili, C. Hartigan, M. J. Thompson, S. Malkani, J. P. Lock, D. M. Harlan, P. Clinton, E. Frank, D. M. Wilson, D. DeSalvo, L. Norlander, T. Ly, B. A. Buckingham, J. Diner, M. Dezube, L. A. Young, A. Goley, M. S. Kirkman, J. B. Buse, H. Zheng, R. R. Selagamsetty, E. R. Damiano, S. J. Russell, Home use of a bihormonal

bionic pancreas versus insulin pump therapy in adults with type 1 diabetes: A multicentre randomised crossover trial. *Lancet* **389**, 369–380 (2017). doi:10.1016/S0140-6736(16)32567-3

173. R. Hovorka, Continuous glucose monitoring and closed-loop systems. *Diabet. Med.* **23**, 1–12 (2006). doi:10.1111/j.1464-5491.2005.01672.x

174. E. Sachs, R.-S. Guo, S. Ha, A. Hu, On-line process optimization and control using the sequential design of experiments. In *Symposium on VLSI Technology* (1990), pp. 99–100. doi:10.1109/vlsit.1990.111027

175. Y. Wang, F. Gao, F. J. Doyle III, Survey on iterative learning control, repetitive control, and run-to-run control. *J. Process Contr.* **19**, 1589–1600 (2009). doi:10.1016/j.jprocont.2009.09.006

176. C. Toffanin, A. Sandri, M. Messori, C. Cobelli, L. Magni, Automatic adaptation of basal therapy for type 1 diabetic patients: a run-to-run approach. *IFAC Proc. Vol.* **47**, 2070–2075 (2014). doi:10.3182/20140824-6-ZA-1003.02462

177. C. Toffanin, M. Messori, C. Cobelli, L. Magni, Automatic adaptation of basal therapy for type 1 diabetic patients: A run-to-run approach. *Biomed. Signal Process. Control* **31**, 539–549 (2017). doi:10.1016/j.bspc.2016.09.002

178. C. Owens, H. Zisser, L. Jovanovič, B. Srinivasan, D. Bonvin, F. J. Doyle III, Run-to-run control of blood glucose concentrations for people with type 1 diabetes mellitus. *IEEE Trans. Biomed. Eng.* **53**, 996–1005 (2006). doi:10.1109/TBME.2006.872818

179. C. C. Palerm, H. Zisser, L. Jovanovič, F. J. Doyle III, A run-to-run control strategy to adjust basal insulin infusion rates in type 1 diabetes. *J. Process Contr.* **18**, 258–265 (2008). doi:10.1016/j.jprocont.2007.07.010

180. C. C. Palerm, H. Zisser, W. C. Bevier, L. Jovanovič, F. J. Doyle, Prandial insulin dosing using run-to-run control: Application of clinical data and medical expertise to define a suitable performance metric. *Diabetes Care* **30**, 1131–1136 (2007). doi:10.2337/dc06-2115

181. J. Tuo, H. Sun, D. Shen, H. Wang, Y. Wang, Optimization of insulin pump therapy based on high order run-to-run control scheme. *Comput. Methods Programs Biomed.* **120**, 123–134 (2015). doi:10.1016/j.cmpb.2015.04.010

182. C. Toffanin, R. Visentin, M. Messori, F. Di Palma, L. Magni, C. Cobelli, Toward a run-to-run adaptive artificial pancreas: In silico results. *IEEE Trans. Biomed. Eng.* **65**, 479–488 (2018). doi:10.1109/TBME.2017.2652062

183. C. C. Palerm, H. Zisser, L. Jovanovič, F. J. Doyle III, Flexible run-to-run strategy for insulin dosing in type 1 diabetic subjects. *IFAC Proc. Vol.* **39**, 521–526 (2006).

184. C. Palerm, H. Zisser, L. Jovanovič, F. Doyle III, A run-to-run framework for prandial insulin dosing: Handling real-life uncertainty. *Int. J. Robust Nonlinear Control* **17**, 1194–1213 (2007). doi:10.1002/rnc.1103

185. J. B. Lee, E. Dassau, F. J. Doyle III, A run-to-run approach to enhance continuous glucose monitor accuracy based on continuous wear. *IFAC-PapersOnLine* **48**, 237–242 (2015). doi:10.1016/j.ifacol.2015.10.145

186. H. Zisser, C. C. Palerm, W. C. Bevier, F. J. Doyle III, L. Jovanovič, Clinical update on optimal prandial insulin dosing using a refined run-to-run control algorithm. *J. Diabetes Sci. Technol.* **3**, 487–491 (2009). doi:10.1177/193229680900300312

187. J. Kolodner, *Case-Based Reasoning* (Morgan Kaufmann, 2014).

188. M. Reddy, P. Pesl, M. Xenou, C. Toumazou, D. Johnston, P. Georgiou, P. Herrero, N. Oliver, Clinical safety and feasibility of the advanced bolus calculator for type 1 diabetes based on case-based reasoning: A 6-week nonrandomized single-arm pilot study. *Diabetes Technol. Ther.* **18**, 487–493 (2016). doi:10.1089/dia.2015.0413

189. E. M. Aiello, C. Toffanin, M. Messori, C. Cobelli, L. Magni, Postprandial glucose regulation via KNN meal classification in type 1 diabetes. *IEEE Control Syst. Lett.* **3**, 230–235 (2018). doi:10.1109/LCSYS.2018.2844179

190. E. Daskalaki, P. Diem, S. G. Mougiakakou, An actor–critic based controller for glucose regulation in type 1 diabetes. *Comput. Methods Programs Biomed.* **109**, 116–125 (2013). doi:10.1016/j.cmpb.2012.03.002

191. P. D. Ngo, S. Wei, A. Holubová, J. Muzik, F. Godtliebsen, Reinforcement-learning optimal control for type-1 diabetes. In *EMBS International Conference on Biomedical & Health Informatics* (2018), pp. 333–336. doi:10.1109/BHI.2018.8333436

192. F. S. Melo, S. P. Meyn, M. I. Ribeiro, An analysis of reinforcement learning with function approximation. In *International Conference on Machine Learning* (2008), pp. 664–671. doi:10.1145/1390156.1390240

193. L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement learning: A survey. *J. Artif. Intell. Res.* **4**, 237–285 (1996). doi:10.1613/jair.301

194. J. M. Hammersley, Monte Carlo methods for solving multivariable problems. *Ann. N.Y. Acad. Sci.* **86**, 844–874 (1960). doi:10.1111/j.1749-6632.1960.tb42846.x

195. P. S. Thomas, E. Brunskill, Data-efficient off-policy policy evaluation for reinforcement learning. *Proc. Mach. Learn. Res.* **48**, 2139–2148 (2016).

196. D. M. Maahs, B. A. Buckingham, J. R. Castle, A. Cinar, E. R. Damiano, E. Dassau, J. H. DeVries, F. J. Doyle III, S. C. Griffen, A. Haidar, L. Heinemann, R. Hovorka, T. W. Jones, C. Kollman, B. Kovatchev, B. L. Levy, R. Nimri, D. N. O'Neal, M. Philip, E. Renard, S. J. Russell, S. A. Weinzimer, H. Zisser, J. W. Lum, Outcome measures for

artificial pancreas clinical trials: A consensus report. *Diabetes Care* **39**, 1175–1179 (2016). doi:10.2337/dc15-2716

197. C. Ellingsen, E. Dassau, H. Zisser, B. Grosman, M. W. Percival, L. Jovanovič, F. J. Doyle III, Safety constraints in an artificial pancreatic β cell: An implementation of model predictive control with insulin on board. *J. Diabetes Sci. Technol.* **3**, 536–544 (2009). doi:10.1177/193229680900300319

198. C. Toffanin, H. Zisser, F. J. Doyle III, E. Dassau, Dynamic insulin on board: Incorporation of circadian insulin sensitivity variation. *J. Diabetes Sci. Technol.* **7**, 928–940 (2013). doi:10.1177/193229681300700415

199. G. D. Konidaris, S. Osentoski, P. S. Thomas, Value function approximation in reinforcement learning using the Fourier basis. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence* (2011), pp. 380–395; www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3569/3885.

200. A. T. Høstmark, G. S. Ekeland, A. C. Beckstrøm, H. D. Meen, Postprandial light physical activity blunts the blood glucose increase. *Prev. Med.* **42**, 369–371 (2006). doi:10.1016/j.ypmed.2005.10.001

201. Z. Guo, P. S. Thomas, E. Brunskill, Using options and covariance testing for long horizon off-policy policy evaluation. *Adv. Neural Inform. Process. Syst.* **30**, 2492–2501 (2017).

202. Q. Liu, L. Li, Z. Tang, D. Zhou, Breaking the curse of horizon: Infinite-horizon off-policy estimation. *Adv. Neural Inform. Process. Syst.* **31**, 5356–5366 (2018).

203. J. P. Hanna, S. Niekum, P. Stone, Importance sampling policy evaluation with an estimated behavior policy. *Proc. Mach. Learn. Res.* **97**, 2605–2613 (2019).

204. D. S. Brown, S. Niekum, Toward probabilistic safety bounds for robot learning from demonstration. In *2017 AAAI Fall Symposium Series* (2017), pp. 10–18; https://aaai.org/ocs/index.php/FSS/FSS17/paper/view/16023/15282.

205. S. Kakade, Optimizing average reward using discounted rewards. In *Annual Conference on Computational Learning Theory* (2001), pp. 605–615. doi:10.1007/3-540-44581-1_40

206. L. Bottou, J. Peters, J. Quiñonero-Candela, D. X. Charles, D. M. Chickering, E. Portugaly, D. Ray, P. Simard, E. Snelson, Counterfactual reasoning and learning systems: The example of computational advertising. *J. Mach. Learn. Res.* **14**, 3207–3260 (2013).

207. F. Berkenkamp, A. Krause, A. P. Schoellig, Bayesian Optimization with Safety Constraints: Safe and Automatic Parameter Tuning in Robotics. arXiv 1602.04450 [cs.RO] (14 February 2016).

208. E. Delage, Y. Ye, Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Oper. Res.* **58**, 595–612 (2010). doi:10.1287/opre.1090.0741

209. Z. Wang, P. W. Glynn, Y. Ye, Likelihood robust optimization for data-driven problems. *Comput. Manage. Sci.* **13**, 241–261 (2016). doi:10.1007/s10287-015-0240-3

210. P. M. Mohajerin Esfahani, D. Kuhn, Data-driven distributionally robust optimization using the Wasserstein metric: Performance guarantees and tractable reformulations. *Math. Program.* **171**, 115–166 (2018). doi:10.1007/s10107-017-1172-1

211. C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, A. Roth, The reusable holdout: Preserving validity in adaptive data analysis. *Science* **349**, 636–638 (2015). doi:10.1126/science.aaa9375

212. K. Gourgoulias, M. A. Katsoulakis, L. Rey-Bellet, J. Wang, How biased is your model? Concentration Inequalities, Information and Model Bias. arXiv 1706.10260 [cs.IT] (30 June 2017).

213. G. Katz, C. Barett, D. L. Dill, K. Julian, M. J. Kochenderfer, Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification* (2017), pp. 97–117. doi:10.1007/978-3-319-63387-9_5

214. S. R. Howard, A. Ramdas, J. McAuliffe, J. Sekhon, Uniform, nonparametric, non-asymptotic confidence sequences. arXiv 1810.08240 [math.ST] (18 October 2018).