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: $\Theta \rightarrow \mathbb{R}$. 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: $\Theta \rightarrow \mathbb{R}$ 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: $\mathcal{A} \rightarrow \mathbb{R}$ 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.

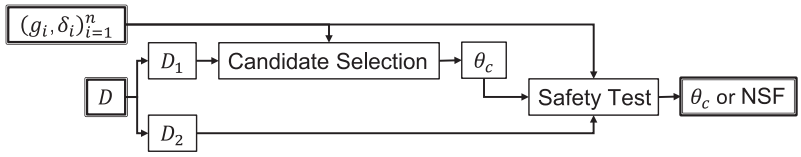

Fig. 1. Overview of Seldonian regression algorithms. The algorithm takes the behavioral constraints $(g_i, \delta_i)_{i=1}^n$ and training data D as input and outputs either a solution θ_c or NSF (no solution found). First, the data are partitioned into two sets, D_1 and D_2. Next, a routine called Candidate 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).

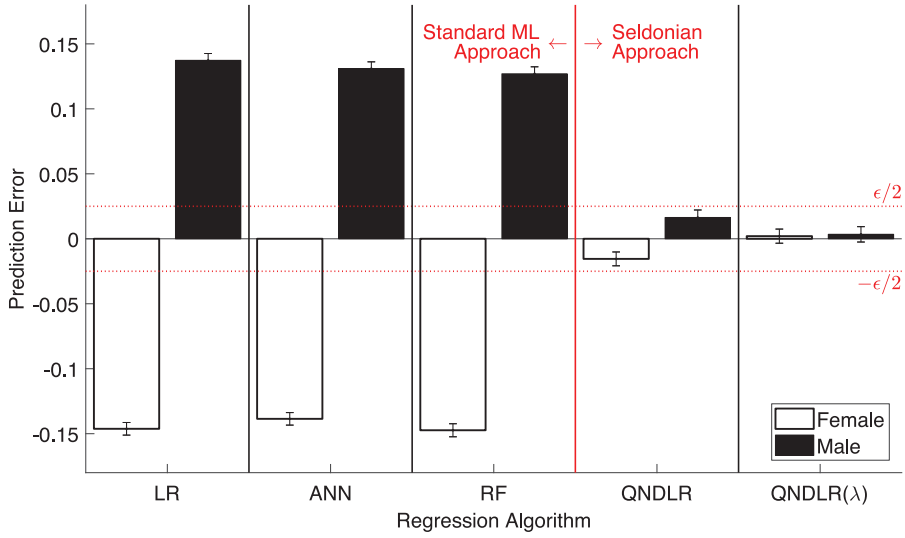

Fig. 2. Seldonian regression algorithm applied to GPA prediction. We used five different regression algorithms to predict students' GPAs during their first three semesters at university based on their scores on nine entrance exams. We used actual data from 43,303 students from Brazil. Here, the user-selected 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

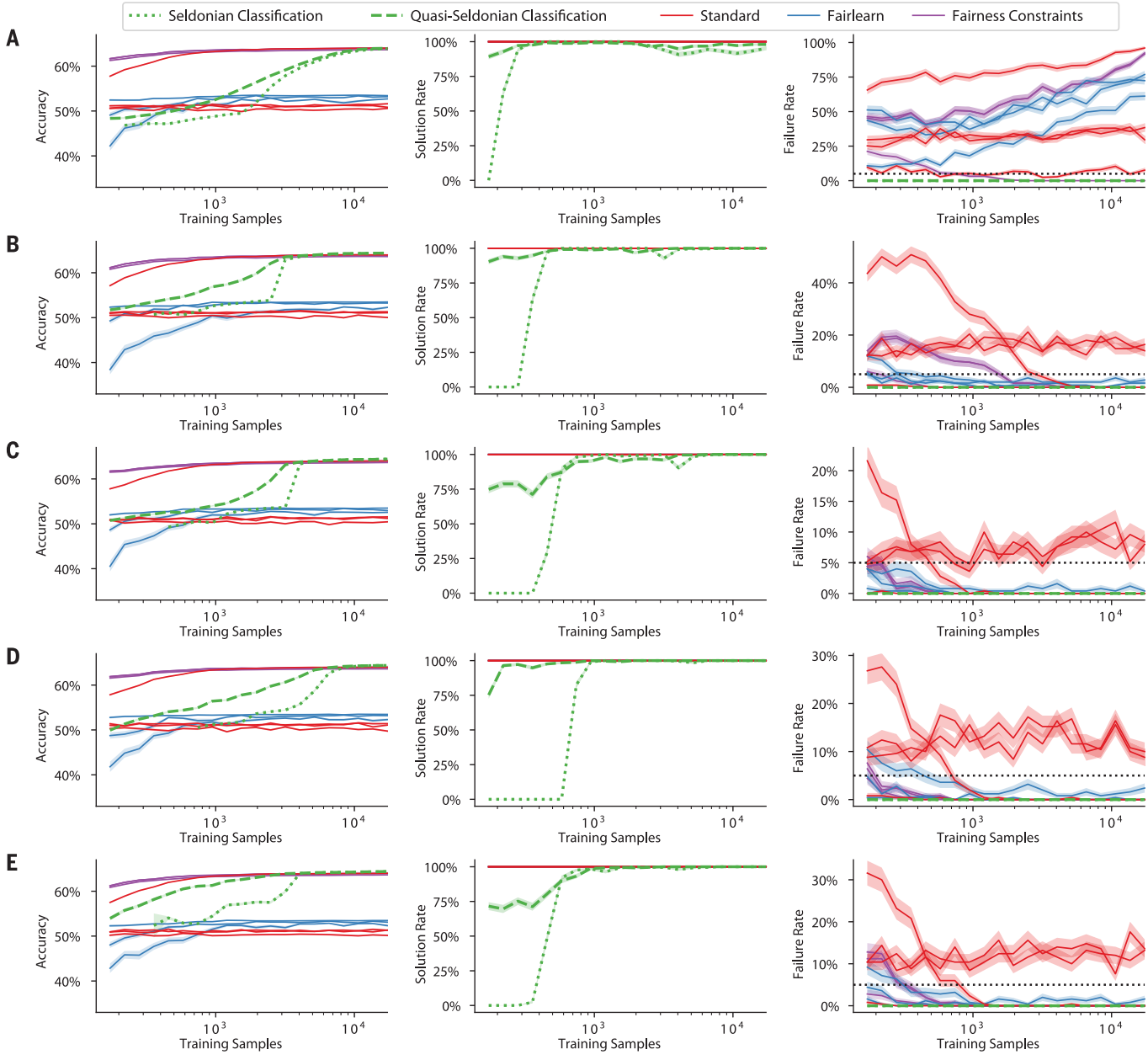

Fig. 3. Seldonian classification algorithm applied to GPA prediction. We applied using classification algorithms to predict whether student GPAs will be above 3.0. Shaded regions represent SE over 250 trials. The curves labeled "Standard" correspond to common classification algorithms designed using 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

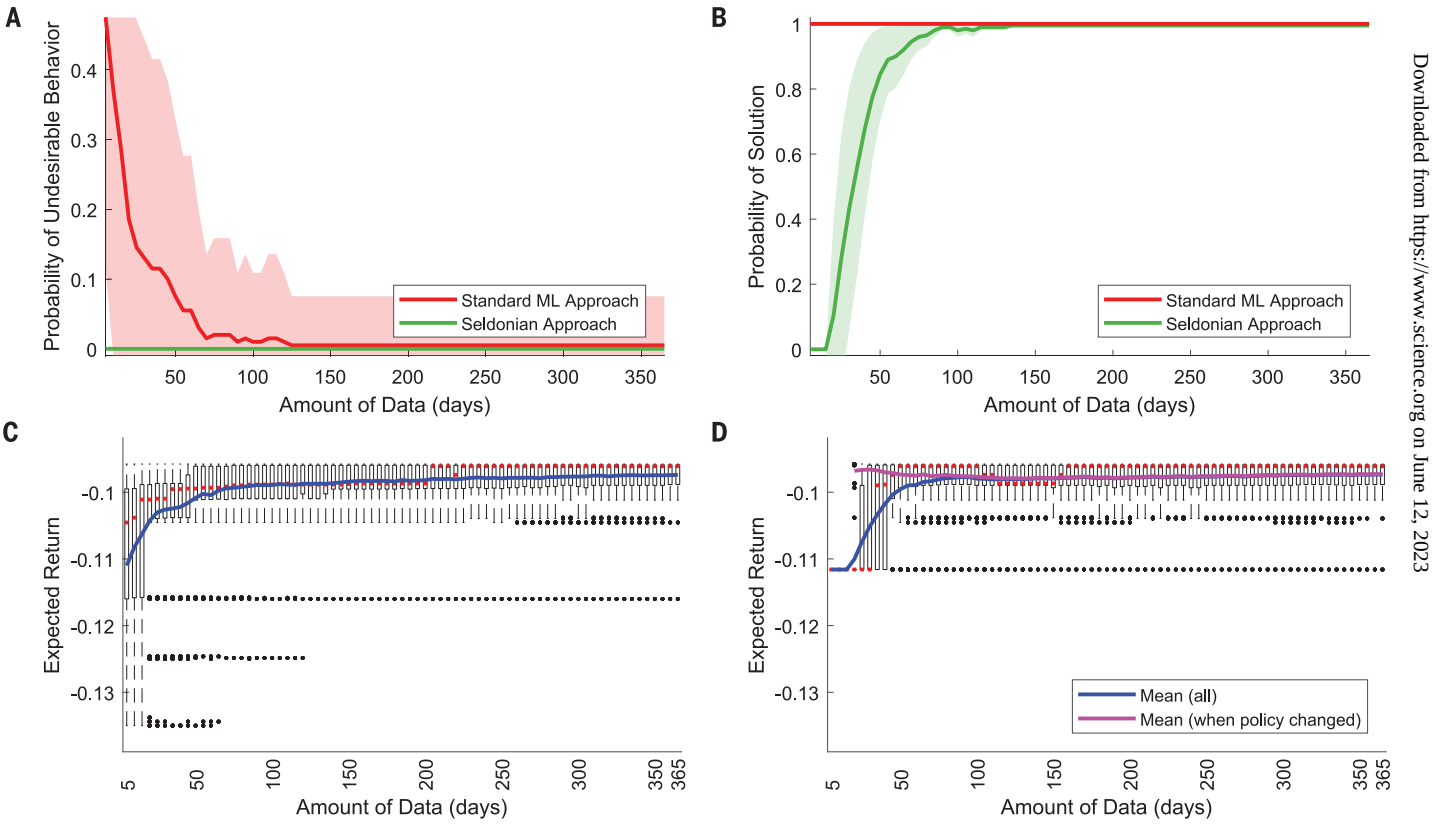

Fig. 4. Seldonian reinforcement learning algorithm for proof-of-principle bolus calculation in type 1 diabetes. Results are averaged over 200 trials; shaded regions denote SE. The Seldonian algorithm is compared to an algorithm built using the standard ML approach that penalizes the prevalence of low 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), .
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