Characterizing and Improving the Robustness of Predict-Then-Optimize Frameworks

Sonja Johnson-Yu[1 [0009−0000−8540−452X]], Jessie Finocchiaro[1,2 [0000−0002−3222−0089]], Kai Wang[3 [2222−3333−4444−5555]], Yevgeniy Vorobeychik[5 [0000−0003−2471−5345]], Arunesh Sinha[6 [0000−0002−3594−3848]], Aparna Taneja[4 [0009−0006−9693−487X]], and Milind Tambe[1,2,4 [0000−0003−3296−3672]]

[1] Harvard University, Cambridge MA, USA {sjohnsonyu@g., jessie@seas., tambe@g.}harvard.edu
[2] Center for Research on Computation and Society
[3] MIT kaiwa318@mit.edu
[4] Google Research aparnataneja@google.com
[5] Washington University St. Louis vorobeychik@wustl.edu
[6] Rutgers University arunesh.sinha@rutgers.edu

Abstract. Optimization tasks situated in incomplete information settings are often preceded by a prediction problem to estimate the missing information; past work shows the traditional predict-then-optimize (PTO) framework can be improved by training a predictive model with respect to the optimization task through a PTO paradigm called *decision-focused learning*. Little is known, however, about the performance of traditional PTO and decision-focused learning when exposed to *adversarial label drift*. We provide modifications of traditional PTO and decision-focused learning that attempt to improve robustness by anticipating label drift. When the predictive model is perfectly expressive, we cast these learning problems as Stackelberg games. With these games, we provide a necessary condition for when anticipating label drift can improve the performance of a PTO algorithm: if performance can be improved, then the downstream optimization objective must be *asymmetric*. We then bound the loss of decision quality in the presence of adversarial label drift to show there may exist a strict gap between the performance of the two algorithms. We verify our theoretical findings empirically in two asymmetric and two symmetric settings. These experimental results demonstrate that robustified decision-focused learning is generally more robust to adversarial label drift than both robust and traditional PTO.

Keywords: predict-then-optimize · adversarial label drift · decision-focused learning

1 Introduction

The *predict-then-optimize* (PTO) framework [4, 11, 14, 24, 33] is a common paradigm for making "smart" decisions with incomplete information. In this framework, one *predicts* information about the state of the world, and then

optimizes a given reward function based on these predictions, possibly subject to constraints. One example of PTO is the problem of hospital bed demand prediction during the COVID-19 pandemic, when hospitals faced shortages of beds but were able to acquire overflow spaces for patients at an additional cost [17, 27]. These hospitals needed to *predict* the true distribution of hospital beds needed and then choose the *optimal* number of overflow beds to purchase [13, 20]. For example, one may want to predict the distribution of bed demand, then optimize to order beds such that there are enough beds on 95% of nights. The traditional way of doing this is to train a *two-stage* (TS) model in which one first learns a predictive model to maximize predictive accuracy, and then runs an optimization algorithm maximizing a *decision quality function* over the trained model's predictions. In this paradigm, predictive accuracy is not always aligned with the reward function being optimized [9]. To correct for this, *decision-focused learning* (DFL) [14, 33], in contrast to the traditional PTO paradigm of two-stage learning (TS), trains the predictive model to directly optimize the decision quality function and differentiates through the entire prediction and optimization pipeline, making training an end-to-end process. DFL has been shown to outperform TS across a wide variety of domains [9, 22, 28, 32].

While predict-then-optimize frameworks are becoming increasing popular, their *robustness* to label drift at decision time is not well-understood. Understanding robustness enables us to reason about the suitability of the models for various test sets. Differences between labels in train and test sets is common in practice. In hospital bed demand prediction, this difference in labels might arise if the prognosis of different viral variants changes. Our primary method of assessing a model's ability to deal with such differences between training and testing sets is by adding perturbations to the underlying labels at test time and assessing how it impacts the model's downstream performance. Specifically, we study *adversarial* label perturbations, with the intuition that a model that can handle worst-case label drift is robust. This label drift can have a profound effect on the downstream decisions, leading to, for example, a shortage of hospital beds. Before decision-focused learning becomes more widely applied in real-world settings, it is imperative to understand the robustness of predict-then-optimize algorithms to label drift.

Our contributions include the following: First, we propose modifications of both TS and DFL that anticipate label noise in training (Robust TS, Robust DFL, and Algorithm 1). Next, by examining the decision quality function, we give a necessary condition for when a learner can improve performance by anticipating label drift: the decision quality function must be *asymmetric* around the optimal decision (Theorem 1). Moreover, we derive bounds on their relative performance by casting both optimization problems as Stackelberg games (§ 4.2) to demonstrate that *Robust DFL is at least as robust as Robust TS* when the predictive model is perfectly expressive. Finally, we empirically validate these theoretical results by comparing Robust DFL and Robust TS across four domains (§ 5) and empirically demonstrate that Robust DFL outperforms Robust TS under adversarial label drift.

1.1 Related Work

Predict-then-optimize and decision-focused learning Predictive models maximizing accuracy on the entire dataset sometimes do so with the consequence of making suboptimal decisions with those predictions in hand (cf. [6, 15]). This suboptimal decision-making is partially remedied by decision-focused learning, which integrates the downstream decision-making objective into the learning pipeline. Decision-focused learning has been applied in settings where the optimization task is combinatorial in nature [3, 22, 33]. Kotary et al. [19] surveys recent efforts to leverage ML to solve constrained optimization problems.

Robustness and optimization in machine learning Our work falls in the intersection of work on decision-focused learning and the broad literature on adversarial machine learning [12, 31]. Much of this literature considers label tampering, which is commonly studied in the context of data poisoning attacks, in which labels in a training dataset are adversarially changed prior to model learning. For example, Butler et al. [8] demonstrate that predict-then-optimize frameworks are susceptible to data poisoning attacks on data features in the training set. In contrast, we are concerned with adversarial drift on labels at test time– after the model has been trained. The literature focusing on adversarial drift at test time [10, 16, 30] is largely concerned with shifts in features rather than labels, often using robust optimization [21, 35] to these shifts in features. A related problem is that of adversarial label contamination, where a small number of labels in the training set are flipped [34]. Also connected is H-infinity control [5], which minimizes the gain in the system states with respect to bounded noise.

2 Background

2.1 Predict-Then-Optimize Problems

In the standard predict-then-optimize framework, one makes a prediction about the state of the world, then optimizes a decision quality function given the prediction. The predictive task is to learn a parameterized (by w) model $m_w : \mathcal{X} \to \mathcal{Y}$ mapping features $x \in \mathcal{X}$ to predict the unknown parameters $m_w(x) = \hat{y} \in \mathcal{Y} \subseteq \mathbb{R}^d$ in the optimization problem. This prediction is used to make decisions based on a *decision quality function* $f : \mathcal{Z} \times \mathcal{Y} \to \mathbb{R}$ mapping decisions $z \in \mathcal{Z} \subseteq \mathbb{R}^\ell$ and true labels $y \in \mathcal{Y}$ to a real-valued reward. Given a prediction $\hat{y}$, the learner computes the optimal decision $z^*(\hat{y})$, where the function $z^* : \mathcal{Y} \to \mathcal{Z}$ is defined

$$z^*(\hat{y}) \coloneqq \arg\max_{z \in \mathcal{Z}} f(z, \hat{y}) \ . \tag{1}$$

The decision is then evaluated on the ground truth label y_0 to obtain the decision quality $f(z^*(\hat{y}), y_0)$. The learner is given a dataset $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}$ to train the predictive model. After the model m_w is trained, a testing dataset $\mathcal{D}_{\text{test}}$ is presented. The trained model then yields predictions of the missing labels and propose the corresponding decisions. The decisions are evaluated on the revealed ground truth labels in the testing set $\frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(x,y) \in \mathcal{D}_{\text{test}}} f(z^*(m_w(x)), y)$.

Assumptions on f Throughout, we generally assume that f is L-Lipschitz continuous in both $\mathcal{Y}$ and $\mathcal{Z}$ and its maximum is always attained by some $z \in \mathcal{Z}$ so that $z^*(\cdot)$ is always well-defined. Moreover, we assume that $f(z^*(\cdot), \cdot)$ is quasi-concave-convex, meaning that $f(z^*(\cdot), y)$ is quasiconvex for all $y \in \mathcal{Y}$ and $f(z^*(\hat{y}), \cdot)$ is quasiconcave for all $\hat{y} \in \mathcal{Y}$. This assumption is necessary in order to apply the minimax theorem in § 4.2. If z^* is affine in y, this condition is satisfied by f being quasi-concave-convex.

2.2 Frameworks for Predict-Then-Optimize

We now summarize two existing learning methods with different objectives that the learner uses to train the predictive model m_w. These are visualized in Fig. 1(L). When unclear from context, we refer to TS and DFL as "Standard TS" and "Standard DFL" to disambiguate them from their robust counterparts introduced in § 3.

The two-stage (TS) approach learns a predictive model m_w by minimizing root mean squared error (RMSE):

$$\min_{w} \quad \sum_{(x,y)\in\mathcal{D}_{\text{train}}} \|m_w(x) - y\|^2 \quad , \tag{Standard TS}$$

where the norm $\|\cdot\|$ denotes the Euclidean norm. We denote the model learned by Standard TS on the training data $\mathcal{D}_{\text{train}}$ by m_T. After the model is learned, given a new input x, the prediction $y_T = m_T(x)$ is then used to optimize the decision quality function $z^*(y_T)$ in Eq. (1).

In contrast, the objective in decision-focused learning is the decision quality function instead of RMSE.

$$\max_{w} \quad \sum_{(x,y)\in\mathcal{D}_{\text{train}}} f(z^*(m_w(x)), y) \tag{Standard DFL}$$

Similarly, we call the Standard DFL model m_D learned via training data $\mathcal{D}_{\text{train}}$. The advantage of decision-focused learning is the alignment of the training objective and the testing objective f. To optimize the objective in Standard DFL, it is common to use gradient descent, which requires backpropagating through the optimal decision z^* from Eq. (1). This can be achieved by differentiating through the optimality and KKT conditions (cf. [1, 2]).

3 Robust algorithms anticipating worst-case label drift

The dependence of optimization on prediction renders Predict-then-Optimize frameworks particularly vulnerable to shifts in the (optimized-upon) labels at test time. *Adversarial training* is often used in the adversarial machine learning literature to mitigate susceptibility to changes in data at test time. It is natural to ask when adversarial training, or some variant thereof, might improve the decision quality of predict-then-optimize frameworks. We consider the case where

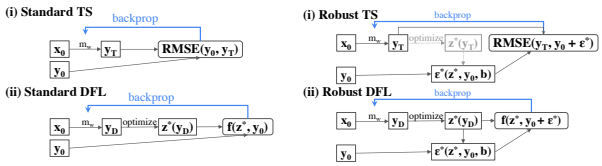

Fig. 1: In TS (left, i), the model optimizes RMSE loss, as opposed to DFL (left, ii), which uses the decision quality function as the loss. Robust analogs (right) are proposed in § 3.

models m_T and m_D are fully expressive and there is no generalization error, and empirically validate that the intuition from that setting still holds in imperfect generalization settings in § 5.

3.1 Modeling Worst-Case Label Drift

We study the robustness of a model by examining its decision quality under adversarial label drift; intuitively, adversarial drift yields a worst-case decision quality (under some "drift budget") to stress-test a model's robustness.

In understanding the worst case noise, suppose that an "adversary" (*abstraction* for nature generating worst-case label drift) can additively perturb the true parameters y by some small ϵ such that $\|\epsilon\| \leq r$ for a fixed budget r at test time. We assume the "adversary" seeks to choose a best response

$$\epsilon \in \epsilon^*(z, y, r) := \underset{\epsilon : \|\epsilon\| \leq r}{\arg\min}\, f(z, y + \epsilon) \tag{2}$$

to minimize the predictive model's decision quality function given the decision z and true parameters y. If $\epsilon^*(z, y, r)$ is not uniquely determined, we slightly abuse notation and take $\epsilon^*(z, y, r)$ to be any choice in the argmin of Eq. (2), and if r is understood from context, we omit it as an argument. We often study $\epsilon_0 \in \epsilon^*(z^*(y_0), y_0, r)$: an optimal response to the optimal decision.

Observe that the "adversary" seeks to minimize the decision quality function even if the predictive model is optimizing root mean squared error, as in two-stage learning. This is because we are concerned with the downstream decisions recommended by the optimization problem, whose quality is measured by f.

3.2 Improving Decision Quality by Anticipating Label Drift: Motivating Examples

We consider two different decision quality functions where either (A) the optimal decision is the same in the presence and absence of noise, or (B) a learner can make a "smarter" decision by anticipating the presence of noise. We leverage insight from (B) to propose incorporate adversarial training into DFL and TS in § 3.3.

Consider a **quadratic decision quality** function pictured in Fig. 2(L), with $\mathcal{Z} = \mathcal{Y} = \mathbb{R}$. In this case, the optimal decision $z^*(\hat{y}) = \hat{y}$. The leader, without anticipating label drift, maximizes their utility by selecting $\hat{y} = y_0$. If they plan for label drift, then the leader can generally choose some $\hat{y} \neq y_0$ or choose $\hat{y} = y_0$. If $\hat{y} \neq y_0$, then $\epsilon^*(\hat{y}, y_0) = \{r\,\mathbf{sign}(y_0 - \hat{y})\}$, resulting in decision quality $1-(\hat{y}-(y_0+\epsilon^*))^2$ strictly less than $1-r^2$. In contrast, if $\hat{y} = y_0$, then $\epsilon^*(y_0, y_0) = \{-r, r\}$, resulting in decision quality $1 - r^2$. Therefore, a learner cannot benefit from anticipating label noise, as their "best" decision in either case (anticipating label noise or not) is to predict $\hat{y} = y_0$.

In contrast, consider the **"asymmetric" decision quality** function in Fig. 2(R), where, again, $\mathcal{Z} = \mathcal{Y} = \mathbb{R}$ and $z^*(\hat{y}) = \hat{y}$ for all $\hat{y} \in \mathcal{Y}$. As above, suppose a learner anticipates adversarial label drift. Broadly, the learner could choose $\hat{y} < y_0$, $\hat{y} = y_0$ (with $\epsilon^*(y_0, y_0) = \{-r\}$), or $\hat{y} > y_0$ (again, with $\epsilon^*(\hat{y}, y_0) = \{-r\}$). Fig. 2(R; red) demonstrates that choosing $\hat{y} = y_0$ leads to a poor decision quality once the the adversarial label perturbation is added. Therefore, the best decision for a learner anticipating adversarial label drift is some $\hat{y} < y_0$ such that $\epsilon^*(\hat{y}, y_0) = \{r\}$, which mitigates the adversary's attack (yellow). Therefore the learner is able to leverage the asymmetry of the decision quality function in order to choose a "smarter" decision than simply predicting y_0, as they would do without anticipating noise.

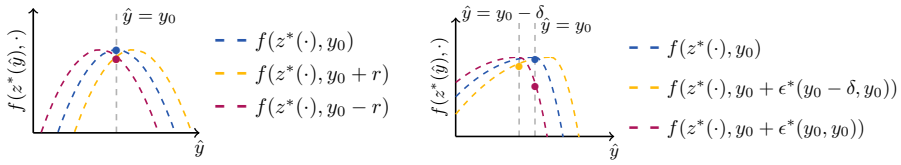

Fig. 2: Two contrasting decision quality functions, both with $z^*(\hat{y}) = \hat{y}$. Different dashed lines represent how the function shifts as y_0 shifts. (L) The optimal decision $z^*(y_0)$ also maximizes the decision quality in the presence of an adversary. Note that $\epsilon^*(y_0, y_0) = \{r, -r\}$, and we demonstrate the decision quality under either adversarially optimal drift (yellow, red lines). (R) The optimal decision in the presence of an adversary is to choose $\hat{y} < y_0$ (yellow dot), as $\epsilon^*(\hat{y}, y_0) = \{r\}$ for such $\hat{y}$. This is in contrast with $\epsilon^*(y_0, y_0) = \{-r\}$. This change in optimal drift leads one to observe that anticipating noise (yellow dot) yields higher decision quality than not anticipating noise (red dot).

3.3 Robust Model Formulations

Observing that some models can improve their performance by anticipating label noise, we now propose robust formulations of both TS and DFL, when the learner anticipates label drift in the test set for such decision quality functions. The learner commits to a model m_w, and applies the model to every instance in

the dataset $(x_0, y_0) \in \mathcal{D}_{\text{train}}$ to produce a prediction $\hat{y} = m_w(x_0)$ and decision $\hat{z} = z^*(\hat{y})$. The "adversary", who knows the true parameters y_0, observes the chosen decision $\hat{z}$ to conduct an attack $\epsilon^*(\hat{z}, y_0, r)$ as defined in Eq. (2).

Definition 1 (Robust two-stage learning). *The learner aims to minimize the root mean squared error between the predictions and the perturbed labels (Fig. 1 (R,i)):*

$$\begin{aligned} \min_{w} \quad & \sum_{(x_0,y_0)\in\mathcal{D}_{train}} \|m_w(x_0) - (y_0 + \epsilon^*_{x_0,y_0})\|^2 \qquad \text{(Robust TS)} \\ s.t. \quad & \epsilon^*_{x_0,y_0} = \underset{\epsilon:\|\epsilon\|\le r}{\arg\min} \quad f(z^*(m_w(x_0)), y_0 + \epsilon) \end{aligned}$$

If the function class given by weights w has high capacity, then we abstract away the model weights and, given any data point (x_0, y_0), we suppose that the model is able to choose $y_{RT} \coloneqq m_{RT}(x_0)$ that is a solution of the following:

$$\max_{y_{RT}} \quad 1 - \|y_{RT} - (y_0 + \epsilon_{RT})\|^2 \quad \text{s.t.} \quad \epsilon_{RT} = \underset{\epsilon:\|\epsilon\|\le r}{\arg\min} f(z^*(y_{RT}), y_0 + \epsilon) \qquad (3)$$

Observe that Robust TS is an analogue of standard adversarial training algorithms, where adversarial changes are now made to labels instead of features. While an alternative formulation might use perturbations that attack the RMSE loss instead of the decision quality, this approach disembodies the robust learning problem from the Predict-Then-Optimize context. Because the algorithm is exposed to adversarial label drift at test time, our Robust TS algorithm is best trained with label drift. We find empirical support for Robust TS in § 5.

Definition 2 (Robust decision-focused learning). *The learner aims to maximize the decision quality evaluated on the perturbed labels (Fig. 1 (R, ii)):*

$$\begin{aligned} \max_{w} \quad & \sum_{(x_0,y_0)\in\mathcal{D}_{train}} f(z^*(m_w(x_0)), y_0 + \epsilon^*_{x_0,y_0}) \qquad \text{(Robust DFL)} \\ s.t. \quad & \epsilon^*_{x_0,y_0} \in \underset{\epsilon:\|\epsilon\|\le r}{\arg\min} \quad f(z^*(m_w(x_0)), y_0 + \epsilon) \end{aligned}$$

Similarly to Robust TS, if the model is fully expressive, then the model can find $y_{RD} \coloneqq m_D(x_0)$ for each (x_0, y_0) pair independently of other data points. This prompts us to solve Robust DFL:

$$\max_{y_{RD}} f(z^*(y_{RD}), y_0 + \epsilon_{RD}) \qquad \text{s.t.} \quad \epsilon_{RD} = \underset{\epsilon:\|\epsilon\|\le r}{\arg\min} \quad f(z^*(y_{RD}), y_0 + \epsilon) \qquad (4)$$

One can understand Equation (4) as a zero-sum game by observing $\epsilon_{RD} \in \arg\max_{\|\epsilon\|\le r} -f(z^*(y_{RD}), y_0 + \epsilon)$. Moreover, as ϵ_{RD} is a function of y_{RD}, casting as a Stackelberg game is natural, where the "adversary" adding noise follows observing the model's prediction y_{RD}. Therefore, we can study whether or not

the pair (y_{RD}, ϵ_{RD}) is an equilibrium solution. Moreover, if $f(z^*(\cdot), \cdot)$ is quasi-concave-convex, then we can apply the canonical minimax theorem [29] and conclude that (y_{RD}, ϵ_{RD}) is a Nash equilibrium as Eq. (4) is *zero-sum*. Casting this problem as a game is similar to the approach taken by Hardt et al. [18], though their model of noise is anticipating noise on features, rather than labels.

4 Improving Robustness by Anticipating Label Drift

With robust formulations in hand, we now show that if adversarial training can improve the *decision quality regret* of an algorithm when worst-case label drift is added at test time, the optimization objective f must be asymmetric in the parameter space around the optimal decision, demonstrated by studying ϵ^*. Knowing that anticipating label drift can improve decision quality, we discuss how to apply Robust TS and Robust DFL in practice, incorporating max-min optimization into training to be robust to label drift.

4.1 Robustness via Defendability

The examples in § 3.2 develop the intuition that *asymmetry of f around the optimal decision* plays an important role in determining the robustness of the decision quality function when models are perfectly expressive.

Definition 3. *A decision quality function $f : \mathcal{Z} \times \mathcal{Y} \to \mathbb{R}$ is* r-defendable *at $y_0 \in \mathcal{Y}$ if there exists $\hat{y} \in \mathcal{Y}$ such that $f(z^*(\hat{y}), y_0 + \epsilon^*(z^*(\hat{y}), y_0, r)) > f(z^*(y_0), y_0 + \epsilon^*(z^*(y_0), y_0, r))$.*

Defendability is tightly connected to the pair $(y_0, \epsilon^*(z^*(y_0), y_0))$ not being a Nash equilibrium to the problem in Eq. (4), modulo the ability to apply the minimax theorem. We now show that defendability also implies a certain type of asymmetry in the decision quality function, studied through ϵ^*.

Theorem 1. *Let f be a decision quality function such that $f(z^*(\cdot), \cdot)$ is quasi-concave-convex. If f is r-defendable at y_0, then $\epsilon^*(z^*(y_0), y_0, r) \neq \{\epsilon : \|\epsilon\| = r\}$.*

Proof. For contradiction, suppose $\epsilon^*(z^*(y_0), y_0, r) = \{\epsilon : \|\epsilon\| = r\}$. Since ϵ^* is defined as a best response, we immediately have $f(z^*(\hat{y}), y_0) \geq f(z^*(\hat{y}), y_0 + \epsilon^*(\hat{y}, y_0))$. Moreover, since f is defendable, we have $f(z^*(\hat{y}), y_0 + \epsilon^*(\hat{y}, y_0)) > f(z^*(y_0), y_0 + \epsilon^*(y_0, y_0)) = f(z^*(y_0), y_0 + \epsilon^*(\hat{y}, y_0))$. Finally, as f is quasiconvex in its second argument and all budget-exhausting responses belong in $\epsilon^*(y_0, y_0)$, we have $\max(f(z^*(y_0), y_0 + \epsilon^*(\hat{y}, y_0)), f(z^*(y_0), y_0 - \epsilon^*(\hat{y}, y_0))) = f(z^*(y_0), y_0 + \epsilon^*(\hat{y}, y_0)) \geq f(z^*(y_0), y_0)$. Chaining these together,

$$\begin{aligned}
f(z^*(\hat{y}), y_0) &\geq f(z^*(\hat{y}), y_0 + \epsilon^*(\hat{y}, y_0)) && \epsilon^* \text{ is best response} \\
&> f(z^*(y_0), y_0 + \epsilon^*(y_0, y_0)) && \text{defendability} \\
&= f(z^*(y_0), y_0 + \epsilon^*(\hat{y}, y_0)) && \text{assumption on } \epsilon^* \\
&\geq f(z^*(y_0), y_0) && \text{quasiconvexity + assumption on } \epsilon^*,
\end{aligned}$$

which contradicts the optimality of z^*.

Theorem 1 shows that if decision quality can be improved by anticipating label drift, it must be the case that the best response to y_0 must be context-dependent, taking into consideration the shape of f.

In the following section, we will present regret bounds for these robust algorithms. It is important to note, however, that for symmetric decision quality functions, these algorithms yield the same solution pairs as their standard counterparts.

Proposition 1. *Consider decision quality function f and $y_0 \in \mathbb{R}^k$ such that $\epsilon^*(z^*(y_0), y_0, r) = \{\epsilon : \|\epsilon\| = r\}$. Then for any $\epsilon_0 \in \epsilon^*(z^*(y_0), y_0, r)$, the pair (y_0, ϵ_0) is a subgame perfect Nash equilibrium for Standard TS (Eq. (5)), Standard DFL (Eq. (6)), Robust TS (Eq. (3)), and Robust DFL (Eq. (4)).*

Proof. We formally cast standard TS and standard DFL as games when models are perfectly expressive.

$$\max_{y_T} 1 - \|y_T - y_0\|^2 \qquad \text{s.t. } \epsilon^*(y_T, y_0) = \arg\min_{\|\epsilon\| \leq r} f(z^*(y_T), y_0 + \epsilon) \tag{5}$$

Observe that the leader's payoff is independent of the follower's payoff $-f(z^*(y_T), y_0 + \epsilon)$. Similarly, we cast Standard DFL as a game when models are perfectly expressive.

$$\max_{y_D} f(z^*(y_D), y_0) \qquad \text{s.t. } \epsilon^*(y_D, y_0) = \arg\min_{\|\epsilon\| \leq r} f(z^*(y_D), y_0 + \epsilon) \tag{6}$$

For standard TS and DFL, since the leader's payoff is independent of the response, choosing y_T or $y_D = y_0$ maximizes the leader's reward, regardless of the adversary's response. Given the decision y_0, the "adversary" playing $\epsilon_0 \in \epsilon^*(z^*(y_0), y_0, r)$ is a best response to optimize their objective.

In Robust TS, there is always a response $\epsilon \in \epsilon^*(z^*(\cdot), y_0)$ such that $\|\epsilon\| = r$ by quasiconvexity of f in its second argument. The objective then becomes $\max_{y_{RT}} 1 - \|y_{RT} - y_0\|^2 + c^2$ for some $c \geq 0$ by quasiconcavity of $f(z^*(\cdot), \cdot)$ in its first objective (which implies that the best response will not improve decision quality). Regardless of the choice of ϵ, then the optimal decision is to report $y_{RT} = y_0$. Thus, (y_0, ϵ_0) is a subgame perfect Nash equilibrium.

Finally in Robust DFL, the assumption implies that f is not r-defendable at y_0 by Theorem 1. Therefore, (y_0, ϵ_0) is a Nash equilibrium, and in turn, a subgame perfect Nash equilibrium since we can apply the minimax theorem.

Proposition 1 suggests that if the adversary's best response to the learner choosing the optimal y_0 can arbitrarily exhaust the noise budget, the standard and robust algorithms have the same solution. Since Robust TS (Eq. (3)) can be understood as a general-sum Stackelberg game, less is known about the convergence to the equilibria than zero-sum Stackelberg games like Robust DFL (c.f., [7]). This may play a larger role in distinguishing the performance of robust and standard algorithms when models have some generalization error, as we demonstrate in § 5. However, for certain asymmetric decision quality functions, we now show that Robust DFL outperforms Robust TS. This yields a simple check to understand whether a decision quality function is robust to label drift.

4.2 Bounding Decision Quality Regret

We now show that Robust DFL yields decision quality no worse than that of Robust TS (Theorem 2). In Proposition 2, we use the piecewise quadratic decision quality function from § 3.2 to demonstrate the performance gap from Robust TS can be *strictly* worse than that of Robust DFL by considering an asymmetric decision quality function, which we show by proving f is defendable implies a strict regret bound.

We start by defining the decision quality regret as the gap in decision quality from of a predictive model to this optimum. Note that while optimizing the decision quality function is a maximization problem for the model, lowering decision quality regret is better as it measures the error induced by poorly responding to label drift.

Definition 4 (Decision quality regret). *Define the* decision quality regret *of prediction $\hat{y}$ when the ground truth parameter is y_0 with an adversarial perturbation budget r by:*

$$Reg(\hat{y}, y_0; r) = f(z^*(y_0), y_0) - \min_{\epsilon:\|\epsilon\|\leq r} f(z^*(\hat{y}), y_0 + \epsilon)$$

The decision quality regret defined in Def. 4 measures the regret of the (optimal) decision $z^*(\hat{y})$ induced by prediction $\hat{y}$ and the worst-case label deviated up to a perturbation of norm r. We now show that the decision quality regret of the optimal Robust TS prediction y_{RT} as at least as high as that of the optimal Robust DFL prediction y_{RD}.

Theorem 2. *Let $f : \mathcal{Z} \times \mathcal{Y} \to \mathbb{R}$ be a L-Lipschitz decision quality function, and consider some ground truth $y_0 \in \mathcal{Y}$ such that (y_{RD}, ϵ_{RD}) is a solution to Eq. (4) and (y_{RT}, ϵ_{RT}) is a solution to Eq (3). Then*

$$0 \leq Reg(y_{RD}, y_0; r) \leq Reg(y_{RT}, y_0; r) \leq 2L \|y_{RT} - y_0\| + Lr \ .$$

Proof. The optimality of Robust DFL can be written as a maximization problem with a worst-case objective: $y_D = \arg\max_{\hat{y}} \left(\min_{\epsilon:\|\epsilon\|\leq r} f(z^*(\hat{y}), y_0 + \epsilon) \right)$. In contrast, the learner in Robust TS does not maximize the worst-case decision quality, which leads to a prediction y_{TS} with a suboptimal objective. Therefore, we have:

$$\min_{\epsilon:\|\epsilon\|\leq r} f(z^*(y_D), y_0 + \epsilon) \geq \min_{\epsilon:\|\epsilon\|\leq r} f(z^*(y_T), y_0 + \epsilon) \ . \tag{7}$$

By the definition of the decision quality regret, Equation 7 directly implies $\text{Reg}(y_D, y_0; r) \leq \text{Reg}(y_T, y_0; r)$. The remaining inequality can be shown by using Lemma 1 (§ A), which concludes the proof.

Theorem 2 quantifies the source of decision quality regret in terms of $\|\hat{y} - y_0\|$ and r. While Theorem 2 gives both upper and lower bounds on the decision quality regret incurred by Robust TS, it is unclear at first whether the gap is ever strict. In Proposition 2, we show by counterexample that the decision quality regret of the optimal Robust TS solution can be strictly greater than that of the optimal Robust DFL solution.

Algorithm 1 Robust Decision-focused Learning

Input: training set $\mathcal{D}_{\text{train}}$, learning rate α, model m_w, adversarial learning rate α_{adv}, perturbation budget r
for epoch $= 1, 2, \cdots$ and $(x_0, y_0) \in \mathcal{D}_{\text{train}}$ **do**
  Generate prediction $\hat{y} = m_w(x_0)$
  Solve Eq. (1) to get decision $z^* = \arg\max_z f(z, \hat{y})$
  Solve Eq. (2) to get perturbation $\hat{\epsilon} = \epsilon^*(z^*, y_0, r)$
  Compute decision quality $f(z^*, y_0 + \hat{\epsilon})$
  Run approximate gradient ascent $m_w \leftarrow m_w + \alpha \frac{\partial f}{\partial z^*} \frac{\partial z^*}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w}$
end for
Return: predictive model m_w

Proposition 2. *For all $r \in \mathbb{R}_{++}$, there exists a decision quality $f : \mathcal{Z} \times \mathcal{Y} \to \mathbb{R}$ such that $Reg(y_{RD}, y_0; r) < Reg(y_{RT}, y_0; r)$.*

Proof. In the setting where the model is perfectly expressive, the solution to Robust TS (Equation 3) yields $y_{RT} = y_0$ as any deviation yields a stronger attack and increased error. We know from Lemma 2 (§ A) that a decision quality function f is r-defendable at y_0 if and only if there exists $\hat{y} \in \mathcal{Y}$ such that $\text{Reg}(\hat{y}, y_0; r) < \text{Reg}(y_0, y_0; r) = \text{Reg}(y_{RT}, y_0; r)$, where equality follows as $y_{RT} = y_0$. If $y_{RT} = y_0$, this tells us that f is r-defendable iff $\text{Reg}(\hat{y}, y_0; r) < \text{Reg}(y_{RT}, y_0; r)$, thus the gap is strict if f is r-defendable at y_0. Observe that the decision quality function in § 3.2 is r-defendable, so the definition is feasible.

When models are perfectly expressive, Theorem 2 shows that the gap between the Robust DFL and Robust TS is bounded by Lr. Moreover, Proposition 2 shows that the gap in decision quality regret between Robust DFL and Robust TS is strict for r-defendable decision quality functions. Together, these results highlight the benefits of using decision-focused learning in the presence of label drift for r-defendable decision quality functions.

4.3 Robust Algorithms in Practice

In practice, optimizing the robust algorithms is not simple. In particular, given the pair $(\hat{y}, y_0)$, the computation of the decision quality $f(z^*(\hat{y}), y_0 + \hat{\epsilon})$ requires estimating (i) optimal decision $\hat{z} = z^*(\hat{y})$ by the optimization problem in Eq. (1), and (ii) the optimal adversarial perturbation $\hat{\epsilon} = \epsilon^*(\hat{z}, y_0, r)$ defined in Eq. (2). Both of these values are not generally defined with closed-form solutions. In Algorithm 1, we leverage the idea from DFL to differentiate through optimization problems and apply the concept of adversarial training to solve the problem in Def. 2. We solve for $\hat{\epsilon}$ by running projected gradient descent, which we instantiate multiple times. The time complexities of training the models can be found in § B. While both Robust TS and Robust DFL are slower than their standard counterparts due to the optimization required at each iteration to calculate $\hat{\epsilon}$, both algorithms still run in polynomial time.

5 Experiments

5.1 Experimental Domains

We now compare the non-robust learning methods discussed in § 2.2 and the robust learning methods discussed in Defs. 1 and 2. We evaluate the performance of different learning methods in four different domains that deal with optimizing over uncertain, estimated parameters where model expressivity is limited. Both the linear top-k and demand prediction domains have asymmetric decision quality functions, while portfolio optimization and budget allocation have symmetric decision quality functions. The latter three domains are drawn from Shah et al. [28], which we augment with the first domain in order to have an additional asymmetric domain. For detailed descriptions of each domain, see § C.

- **Demand Prediction** Using features x_0 to predict the bed demand y_0, select number of beds $z^*(\hat{y})$ via an asymmetric decision quality f with a preference to overestimating demand.
- **Linear Top-k** Using a linear model to predict the utilities y_0 of d resources from features x_0, where the relationship between features and labels is cubic, select the k resources with the highest utilities.
- **Portfolio Optimization** Predict the next stock price from historical stock prices and then choose a continuous allocation $z \in Z \subseteq [0, 1]^d$ between d stocks [26], maximizing the sum of the immediate net profits and a symmetric quadratic risk penalty term.
- **Budget Allocation** Choose a website on which to run advertisements by predicting the click-through rates on each website for each user and then selecting a set of websites on which to run advertisements to maximize the expected number of users who click on the ad.

The adversarial perturbations added to the labels represents a shift in the bed demand, underlying utilities, stock prices, and click-through rates, respectively. We ran all experiments over 30 different random seeds and present the mean of the performances on the test set. These experiments are run with a predictive model that is not fully expressive; despite this, the results still align with the intuition from our theoretical results.

5.2 Discussion of Empirical Results

In Fig. 3, we plot the effect of the budget for the adversarial perturbations at test-time on the decision quality. The robust algorithms are trained with the same budget as is used at test time, omitting the case where the test budget is 0. For more results on performance when the adversarial budget differs between training and test time, see § C. In Fig. 4, we present the RMSE for each of the algorithms' predictions at test time, varying the adversarial perturbation budget r, in order to show how much generalization error each model incurs. Finally, in Fig. 5, we visualize the test-time predictions from the models in the

Demand Prediction domain to show how DFL and Robust DFL have learned to overestimate in order to maximize the reward, which favors overallocation of beds rather than underallocation.

The results presented in § 3 show that Robust DFL will perform better than Robust TS (and TS) in asymmetric domains under perfect expressiveness, and this is verified in our experimental results in Fig. 3. It is notable and surprising that Standard DFL already performs significantly better than both TS and Robust TS in these domains. Therefore in asymmetric domains, it may be useful to spend effort on using even standard DFL for the sake of robustness, rather than on robust version of TS, which ultimately still delivers lower quality.

Robust DFL also offers some improvements over Standard DFL: reducing variance and improving decision quality in high-noise regimes. For example, Robust DFL has a lower variance than DFL, as demonstrated in the Linear Top-k domain of Fig. 3. This difference in variance is statistically significant for most r in Linear Top-k: $p_{r=0.5} < 10^{-5}$, $p_{r=1} < 10^{-5}$, $p_{r=1.5} < 10^{-5}$, $p_{r=2} < 10^{-5}$. Additionally, in the high-noise regimes ($r = 3, 4, 5$) of the Demand Prediction domain, the decision quality yielded by Robust DFL is higher than that of Standard DFL, and this difference is statistically significant ($p_{r=3} < 0.01$, $p_{r=4} < 0.0001$, $p_{r=5} < 10^{-5}$). In the Demand Prediction domain, both Robust DFL and DFL learn to overestimate the bed demand, as seen in Fig. 5, which is incentivized by the decision quality function. We can see, by comparing the left and right sides of Fig. 5, that the addition of label perturbations at train time allow Robust DFL to anticipate and tolerate label drift of a higher magnitude (whereas DFL is non-adaptive because it is not trained with perturbations).

For perfectly expressive models and symmetric decision quality functions, Proposition 1 suggests that all four algorithms will be the same. **Despite the fact that the predictive model is not perfectly expressive, Budget Allocation (bottom right) still exemplifies this phenomenon**, with the qualification that DFL tends to do slightly better than TS. The Portfolio Optimization domain (bottom left), however, shows an unforeseen result: **the robust algorithms (especially Robust DFL) outperform the standard algorithms.** This demonstrates that, despite the symmetry of the decision quality function implying that TS and DFL are able to learn the optimal solution, the robust algorithms offer a practical improvement in performance when models are not perfectly expressive. Notably, as seen in the third plot of Fig. 4, Robust TS achieves lower RMSE than TS. These results show that robustification can help both symmetric and asymmetric domains.

6 Conclusion

In this work, we study robustness of predict-then-optimize frameworks to characterize when an adversarially trained algorithm might outperform its standard counterpart by anticipating noise in the test set. Leveraging these insights, we propose robust versions of DFL and TS, and we show that Robust DFL outperforms Robust TS when the decision quality function is defendable. Finally,

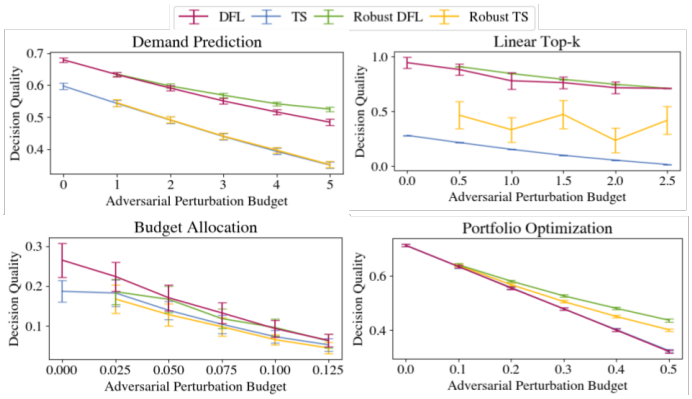

Fig. 3: Effect of adversarial budget on decision quality (30 trials). In the asymmetric Demand Prediction and Linear Top-k domains, DFL and Robust DFL outperform TS and Robust TS.

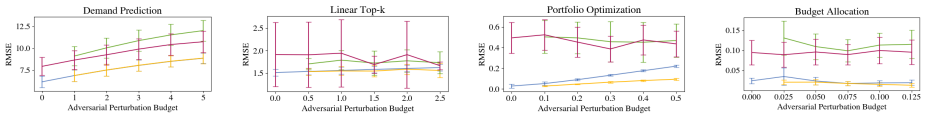

Fig. 4: Effect of adversarial noise budget on RMSE (30 trials). In all domains, DFL/Robust DFL have higher RMSE than TS/Robust TS.

we empirically validate our results with experiments across four domains, finding that Robust DFL does well in asymmetric domains and can even improve performance in symmetric domains.

7 Acknowledgements

Research was sponsored by the ARO and was accomplished under Grant Number: W911NF-18-1-0208. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of ARO or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. Additionally, this material is based upon work supported by the National Science Foundation under Award No. 2202898.

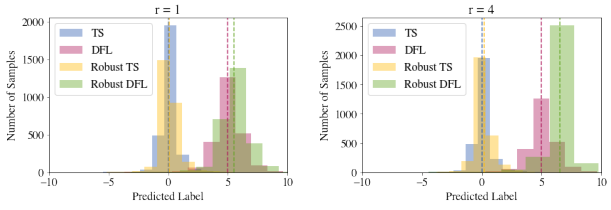

Fig. 5: Predictions from Demand Prediction models, trained on $r = 1$ (L) and $r = 4$ (R) adversarial perturbation budgets.

Bibliography

[1] Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., Kolter, Z.: Differentiable Convex Optimization Layers. arXiv:1910.12430 [cs, math, stat] (Oct 2019), URL http://arxiv.org/abs/1910.12430, arXiv: 1910.12430

[2] Amos, B., Kolter, J.Z.: OptNet: Differentiable Optimization as a Layer in Neural Networks. In: Proceedings of the 34th International Conference on Machine Learning, pp. 136–145, PMLR (Jul 2017), URL https://proceedings.mlr.press/v70/amos17a.html, iSSN: 2640-3498

[3] Amos, B., Koltun, V., Kolter, J.Z.: The limited multi-label projection layer. arXiv preprint arXiv:1906.08707 (2019)

[4] Angalakudati, M., Balwani, S., Calzada, J., Chatterjee, B., Perakis, G., Raad, N., Uichanco, J.: Business analytics for flexible resource allocation under random emergencies. Management Science **60**(6), 1552–1573 (2014)

[5] Başar, T., Bernhard, P.: H-infinity optimal control and related minimax design problems: a dynamic game approach. Springer Science & Business Media (2008)

[6] Beygelzimer, A., Langford, J.: The offset tree for learning with partial labels. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p. 129–138, KDD '09, Association for Computing Machinery, New York, NY, USA (2009), ISBN 9781605584959, https://doi.org/10.1145/1557019.1557040, URL https://doi.org/10.1145/1557019.1557040

[7] Blum, A., Haghtalab, N., Hajiaghayi, M., Seddighin, S.: Computing stackelberg equilibria of large general-sum games. In: International Symposium on Algorithmic Game Theory, pp. 168–182, Springer (2019)

[8] Butler, R., Tuck, W.W., Sinha, A., Ngyuen, T.: Poisoning attacks on data-based decision making: A preliminary study. AASG-22: 3rd Autonomous Agents for Social Good (AASG) held at the 21st International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) (May 2022)

[9] Cameron, C., Hartford, J., Lundy, T., Leyton-Brown, K.: The perils of learning before optimizing. Proceedings of the AAAI Conference on Artificial Intelligence **36**(4), 3708–3715 (Jun 2022), https://doi.org/10.1609/aaai.v36i4.20284, URL https://ojs.aaai.org/index.php/AAAI/article/view/20284

[10] Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A., Kurakin, A.: On evaluating adversarial robustness (2019), https://doi.org/10.48550/arXiv.1902.06705, URL http://arxiv.org/abs/1902.06705

[11] Chan, C.W., Farias, V.F., Bambos, N., Escobar, G.J.: Optimizing intensive care unit discharge decisions with patient readmissions. Operations research **60**(6), 1323–1341 (2012)

[12] Chen, P.Y., Hsieh, C.J.: Adversarial Robustness for Machine Learning. Imprint (2022)

[13] Deschepper, M., Eeckloo, K., Malfait, S., Benoit, D., Callens, S., Vansteelandt, S.: Prediction of hospital bed capacity during the covid- 19 pandemic. BMC health services research **21**(1), 1–10 (2021)
[14] Elmachtoub, A.N., Grigas, P.: Smart "Predict, then Optimize". Management Science **68**(1), 9–26 (Jan 2022), ISSN 0025-1909, https://doi.org/10.1287/mnsc.2020.3922, URL https://pubsonline.informs.org/doi/abs/10.1287/mnsc.2020.3922, publisher: INFORMS
[15] Ford, B., Nguyen, T., Tambe, M., Sintov, N., Fave, F.D.: Beware the soothsayer: From attack prediction accuracy to predictive reliability in security games. In: International Conference on Decision and Game Theory for Security, pp. 35–56, Springer (2015)
[16] Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples (2014), https://doi.org/10.48550/ARXIV.1412.6572, URL https://arxiv.org/abs/1412.6572
[17] Grimm, C.A.: Hospital experiences responding to the covid-19 pandemic: results of a national pulse survey march 23–27, 2020. US Department of Health and Human Services Office of Inspector General **41** (2020)
[18] Hardt, M., Megiddo, N., Papadimitriou, C., Wootters, M.: Strategic classification. In: Proceedings of the 2016 ACM conference on innovations in theoretical computer science, pp. 111–122 (2016)
[19] Kotary, J., Fioretto, F., Hentenryck, P.V., Wilder, B.: End-to-end constrained optimization learning: A survey. CoRR **abs/2103.16378** (2021), URL https://arxiv.org/abs/2103.16378
[20] Kutafina, E., Bechtold, I., Kabino, K., Jonas, S.M.: Recursive neural networks in hospital bed occupancy forecasting. BMC medical informatics and decision making **19**(1), 1–10 (2019)
[21] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. International Conference on Learning Representations (2018)
[22] Mandi, J., Demirovi?, E., Stuckey, P.J., Guns, T.: Smart Predict-and-Optimize for Hard Combinatorial Optimization Problems. Proceedings of the AAAI Conference on Artificial Intelligence **34**(02), 1603–1610 (Apr 2020), ISSN 2374-3468, https://doi.org/10.1609/aaai.v34i02.5521, URL https://ojs.aaai.org/index.php/AAAI/article/view/5521, number: 02
[23] Markowitz, H.M., Todd, G.P.: Mean-variance analysis in portfolio choice and capital markets, vol. 66. John Wiley & Sons (2000)
[24] Mehrotra, M., Dawande, M., Gavirneni, S., Demirci, M., Tayur, S.: Or practice—production planning with patterns: A problem from processed food manufacturing. Operations research **59**(2), 267–282 (2011)
[25] Michaud, R.O.: The markowitz optimization enigma: Is 'optimized'optimal? Financial analysts journal **45**(1), 31–42 (1989)
[26] Popescu, I.: Robust mean-covariance solutions for stochastic optimization. Operations Research **55**(1), 98–112 (2007)
[27] Sen-Crowe, B., Sutherland, M., McKenney, M., Elkbuli, A.: A closer look into global hospital beds capacity and resource shortages during the covid-19 pandemic. journal of Surgical Research **260**, 56–63 (2021)

[28] Shah, S., Wilder, B., Perrault, A., Tambe, M.: Learning (local) surrogate loss functions for predict-then-optimize problems. Neural Information Processing Systems (2022)
[29] Sion, M.: On general minimax theorems. (1958)
[30] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
[31] Vorobeychik, Y., Kantarcioglu, M.: Adversarial machine learning. Synthesis Lectures on Artificial Intelligence and Machine Learning **12**(3), 1–169 (2018)
[32] Wang, K., Verma, S., Mate, A., Shah, S., Taneja, A., Madhiwalla, N., Hegde, A., Tambe, M.: Decision-focused learning in restless multi-armed bandits with application to maternal and child care domain. arXiv preprint arXiv:2202.00916 (2022)
[33] Wilder, B., Dilkina, B., Tambe, M.: Melding the Data-Decisions Pipeline: Decision-Focused Learning for Combinatorial Optimization. Proceedings of the AAAI Conference on Artificial Intelligence **33**(01), 1658–1665 (Jul 2019), ISSN 2374-3468, https://doi.org/10.1609/aaai.v33i01.33011658, URL https://ojs.aaai.org/index.php/AAAI/article/view/3982, number: 01
[34] Xiao, H., Biggio, B., Nelson, B., Xiao, H., Eckert, C., Roli, F.: Support vector machines under adversarial label contamination. Neurocomputing **160**, 53–62 (2015), ISSN 0925-2312, https://doi.org/https://doi.org/10.1016/j.neucom.2014.08.081, URL https://www.sciencedirect.com/science/article/pii/S0925231215001198
[35] Xu, H., Caramanis, C., Mannor, S.: Robustness and regularization of support vector machines. Journal of machine learning research **10**(7) (2009)
[36] Yahoo!: Yahoo! webscope dataset. https://webscope.sandbox.yahoo.com/.ydataysm-advertiser-bids-v1.0 (2007)

A Omitted proofs

Lemma 1. *Let f be L-Lipschitz and fix $r \in \mathbb{R}_+$. For all $y_0, y \in \mathcal{Y}$, $Reg(y, y_0; r) \leq 2L\|y - y_0\| + Lr$*

Proof.

$$\begin{aligned}
\mathrm{Reg}(\hat{y}, y_0; r) &= f(z^*(y_0), y_0) - \min_{\epsilon:\|\epsilon\|\leq r} f(z^*(\hat{y}), y_0 + \epsilon) \\
&= f(z^*(y_0), y_0) - f(z^*(\hat{y}), y_0 + \hat{\epsilon}) \\
&\leq f(z^*(y_0), \hat{y}) + L\,\|y_0 - \hat{y}\| - f(z^*(\hat{y}), \hat{y}) + L\,\|y_0 + \hat{\epsilon} - \hat{y}\| \\
&\leq \underbrace{f(z^*(y_0), \hat{y}) - f(z^*(\hat{y}), \hat{y})}_{\leq 0 \text{ by optimality of } z^*} \qquad +2L\,\|y_0 - \hat{y}\| + Lr \\
&\leq 2L\,\|y_0 - \hat{y}\| + Lr\ .
\end{aligned}$$

Lemma 2. *A decision quality function $f : \mathcal{Z} \times \mathcal{Y} \to \mathbb{R}$ is r-defendable at y_0 if and only if there exists $\hat{y} \in \mathcal{Y}$ such that $Reg(\hat{y}, y_0; r) < Reg(y_0, y_0; r)$.*

Proof. f is r-defendable at y_0 if and only if $\exists \hat{y} \in \mathcal{Y}$ such that

$$\begin{aligned} & -f(z^*(\hat{y}), y_0 + \hat{\epsilon}) < -f(z^*(y_0), y_0 + \epsilon) \\ \iff & f(z^*(y_0), y_0) - f(z^*(\hat{y}), y_0 + \hat{\epsilon}) < f(z^*(y_0), y_0) - f(z^*(y_0), y_0 + \epsilon) \ , \end{aligned}$$

where $\epsilon \coloneqq \epsilon^*(z^*(y_0), y_0, r)$ and $\hat{\epsilon} \coloneqq \epsilon^*(\hat{y}, y_0, r)$.

B Runtime Analysis

TS $= \Theta(T \cdot N \cdot T_M)$, where T is the number of timesteps of model training, N is the number of instances, and T_M is the time to run the forward and backward passes for the predictive model.

DFL $= \Theta(T \cdot N \cdot (T_M + T_Z + T'_Z + T_{DQ} + T'_{DQ}))$, where T_Z is the time for the forward pass of the optimization and T'_Z is the backward pass for the optimization, and T_{DQ}/T'_{DQ} are the forward and backward passes for calculating the decision quality.

Robust TS $= \Theta(T \cdot N \cdot (T_M + T_Z + I \cdot T_A \cdot (T_{DQ} + T'_{DQ})))$, where T_A is the number of iterations to run projected gradient descent, and I is the number of times the perturbation generation process is instantiated (where more instantiations produce a better $\hat{\epsilon}$).

Robust DFL $= \Theta(T \cdot N \cdot (T_M + T_Z + T'_Z + T_{DQ} + T'_{DQ} + I \cdot T_A \cdot (T_{DQ} + T'_{DQ})))$.

C Experimental Setup

We now give an overview of the four studied decision quality functions. Note that we do not enforce the concave-convex assumption on the decision quality functions; rather, the associated experiments show the performance in realistic scenarios where the assumption does not always hold.

Demand Prediction Domain The Demand Prediction task is to predict the number of beds required in a hospital's overflow unit [17, 27]. In this setting, one might have a decision quality with a global maximum at the exact demand but also with a strong preference to overestimating bed demand than underestimating, yielding a decision quality like that in Fig. 6, given in Equation 8.

$$f(z, y) = \frac{1}{1 + e^{-2(z-y+2.73)}} - \frac{1}{0.91(1 + 25e^{-(z-y+2.73)/6})} \tag{8}$$

- **Predict**: Use feature x_0 to predict the demand y_0.
- **Optimize**: Pick a z^* that maximizes the decision quality f in Equation 8.

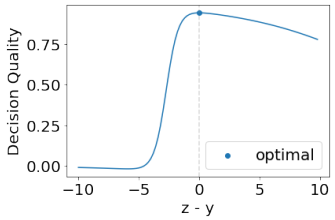

Fig. 6: Decision quality function f for the Demand Prediction domain. The decision quality drops drastically when the demand y is higher than the supply z.

Linear Top-k Domain The linear model domain requires fitting a linear model to data that express a cubic relationship between features and labels. It is drawn from [28], motivated by the importance of such problems in the AI interpretability literature .

- **Predict**: Use feature $x_0 \in \mathbb{R}^d$ to predict the utility $\hat{y} \in \mathbb{R}^d$, where the true utility of resource n is $y_{0_n} = 10{x_0}_n^3 - 6.5x_{0n}$.
- **Optimize**: Pick the top $b = 1$ from d resources, $z^*(\hat{y}) = \arg\text{topk}(\hat{y})$.

Portfolio Optimization Domain The Portfolio Optimization domain is a quadratic programming problem [26], where investors choose a continuous allocation $z \in Z \subseteq [0, 1]^n$ between n stocks, subject to a budget constraint $Z = \{z \in [0, 1]^n : \mathbf{1}^\top z = 1\}$. We implement the Markowitz formulation [23, 25].

- **Predict**: Given d historical stock prices for n stocks, $x \in \mathbb{R}^{n\times d}$, predict the next stock prices $y \in \mathbb{R}^n$.
- **Optimize**: Given n predicted stock prices $\hat{y}$, maximize the weighted sum of the immediate net profit minus a risk penalty term, with $f(z, y) = z^\top y - \lambda z^\top Q z$. We let the risk aversion constant $\lambda = 0.1$ and Q be the identity matrix for simplicity.

Budget Allocation Domain The Budget Allocation domain is a submodular optimization problem adapted from Wilder et al. [33], where the task is to choose a website on which to run advertisements using click-through rates (CTRs) from the Yahoo! Webscope Dataset [36]. This is a particularly difficult problem for Robust DFL due to the combinatorial structure of the optimization problem. The advertisement plan is discrete and it is hard for DFL to differentiate through and learn from discrete decisions.

- **Predict**: Given d features for m websites, $x \in \mathbb{R}^{m\times d}$, predict the CTRs of n users $y \in \mathbb{R}^{m\times n}$.
- **Optimize**: Given the predicted matrix of user/website CTRs $\hat{y}$, select a set of websites (subject to budget constraint r) denoted by $z \in \{0, 1\}^m$ on which to run advertisements to maximize the expected number of users who click on the ad at least once. We also incorporate a weight matrix w that represents the number of times that a user will see an ad on a given website.

$$f(z, y) = \sum_{j=0}^{n}(1 - \prod_{i=0}^{m}(1 - z_i \cdot y_{ij})^{w_{ij}})$$

D Additional empirical analysis

Performance of Robust algorithms with Unknown Noise Budgets In addition to showing that Robust DFL is robust when the adversarial budget is known at training time, we demonstrate that the generalizability of Robust DFL to different noise budgets. In Fig. 7, each row represents a different training noise budget, while each column represents a testing noise budget. Each cell is combination between training noise and testing noise, containing the difference in decision quality between Robust DFL and Robust TS. We observe that Robust DFL outperforms Robust TS in a higher number of train/test noise combinations. In particular, on and above the diagonal where the train noise is closer to the test noise, the improvement is more significant.

Demand Prediction Robust DFL - Robust TS

	test noise budget					
train noise budget	0.0	1.0	2.0	3.0	4.0	5.0
0.0	0.082	0.090	0.100	0.111	0.123	0.134
1.0	0.077	0.089	0.103	0.118	0.128	0.148
2.0	0.076	0.089	0.106	0.124	0.138	0.159
3.0	0.073	0.090	0.107	0.128	0.147	0.164
4.0	0.068	0.086	0.104	0.123	0.145	0.162
5.0	0.064	0.082	0.105	0.124	0.149	0.174

Linear Top-k Robust DFL - Robust TS

	test noise budget					
train noise budget	0.0	0.5	1.0	1.5	2.0	2.5
0.0	0.663	0.666	0.625	0.663	0.661	0.693
0.5	0.449	0.443	0.489	0.549	0.511	0.539
1.0	0.364	0.332	0.512	0.435	0.425	0.507
1.5	0.505	0.390	0.266	0.319	0.435	0.387
2.0	0.424	0.460	0.504	0.470	0.512	0.531
2.5	0.539	0.481	0.513	0.462	0.407	0.291

Portfolio Optimization Robust DFL - Robust TS

	test noise budget					
train noise budget	0.0	0.1	0.2	0.3	0.4	0.5
0.1	-0.007	0.002	0.012	0.018	0.030	0.038
0.2	-0.010	0.000	0.013	0.025	0.034	0.052
0.3	-0.022	-0.007	0.007	0.021	0.038	0.050
0.4	-0.022	-0.013	0.002	0.016	0.030	0.044
0.5	-0.036	-0.018	-0.005	0.011	0.021	0.035

Budget Allocation Robust DFL - Robust TS

train noise budget	test noise budget					
0.0	0.000	0.025	0.050	0.075	0.100	0.125
0.000	0.010	0.006	0.004	0.004	0.003	0.001
0.025	0.005	0.002	0.005	0.002	0.003	0.001
0.050	0.007	0.008	0.005	0.002	0.002	0.001
0.075	0.007	0.006	0.005	0.003	0.002	0.001
0.100	0.009	0.007	0.007	0.004	0.004	0.002
0.125	0.009	0.009	0.005	0.004	0.003	0.002

Demand Prediction Robust DFL - DFL

	test noise budget					
train noise budget	0.0	1.0	2.0	3.0	4.0	5.0
1.0	-0.002	0.001	0.005	0.011	0.009	0.015
2.0	-0.005	0.001	0.006	0.013	0.016	0.025
3.0	-0.007	0.000	0.008	0.018	0.024	0.031
4.0	-0.008	0.000	0.009	0.017	0.026	0.035
5.0	-0.011	-0.001	0.007	0.019	0.028	0.041

Linear Top-k Robust DFL - DFL

	test noise budget					
train noise budget	0.0	0.5	1.0	1.5	2.0	2.5
0.5	-0.027	0.027	0.036	0.001	0.031	-0.063
1.0	0.030	0.028	0.066	-0.062	0.005	0.000
1.5	0.001	0.026	0.039	0.030	0.030	-0.001
2.0	0.029	0.026	0.066	0.030	0.031	0.001
2.5	0.029	0.002	0.067	0.029	0.031	0.001

Portfolio Optimization Robust DFL - DFL

	test noise budget					
train noise budget	0.0	0.1	0.2	0.3	0.4	0.5
0.1	-0.006	0.006	0.019	0.027	0.043	0.054
0.2	-0.012	0.006	0.025	0.042	0.058	0.083
0.3	-0.024	0.000	0.025	0.048	0.075	0.097
0.4	-0.029	-0.005	0.025	0.051	0.080	0.109
0.5	-0.048	-0.011	0.021	0.053	0.083	0.115

Budget Allocation Robust DFL - DFL

train noise budget	test noise budget					
0.000	0.025	0.050	0.075	0.100	0.125	
0.025	-0.003	-0.005	-0.001	-0.001	0.000	-0.001
0.050	-0.002	-0.002	-0.001	-0.002	-0.001	0.000
0.075	-0.003	-0.002	-0.002	-0.002	-0.001	-0.001
0.100	-0.003	-0.003	0.000	-0.001	0.000	0.000
0.125	-0.037	-0.031	-0.024	-0.019	-0.014	0.000

Fig. 7: The shade of teal depicts the extent to which Robust DFL outperforms Robust TS. The improvement of Robust DFL over DFL highlights the importance of using the decision quality and the importance of considering the adversarial perturbation, respectively. Negative cells are where Robust TS outperforms Robust DFL. Intuitively, these regions are where Robust DFL tries to defend against noise that is simply not present at test time.