Leaving the Nest 🦅:
Going Beyond Local Loss Functions for Predict-Then-Optimize

Sanket Shah[1], Bryan Wilder[2], Andrew Perrault[3], Milind Tambe[1]

[1]Harvard University
[2]Carnegie Mellon University
[3]Ohio State University
sanketshah@g.harvard.edu, bwilder@andrew.cmu.edu, perrault.17@osu.edu, milind_tambe@harvard.edu

Abstract

Predict-then-Optimize is a framework for using machine learning to perform decision-making under uncertainty. The central research question it asks is, "How can we use the structure of a decision-making task to tailor ML models for that specific task?" To this end, recent work has proposed learning task-specific loss functions that capture this underlying structure. However, current approaches make restrictive assumptions about the form of these losses and their impact on ML model behavior. These assumptions both lead to approaches with high computational cost, and when they are violated in practice, poor performance. In this paper, we propose solutions to these issues, avoiding the aforementioned assumptions and utilizing the ML model's features to increase the sample efficiency of learning loss functions. We empirically show that our method achieves state-of-the-art results in four domains from the literature, often requiring an order of magnitude fewer samples than comparable methods from past work. Moreover, our approach outperforms the best existing method by nearly 200% when the localness assumption is broken.

1 Introduction

Predict-then-Optimize (PtO) (Donti, Amos, and Kolter 2017; Elmachtoub and Grigas 2021) is a framework for using machine learning (ML) to perform decision-making under uncertainty. As the name suggests, it proceeds in two steps—first, an ML model is used to make predictions about the uncertain quantities of interest, then second, these predictions are aggregated and used to parameterize an optimization problem whose solution provides the decision to be made. Many real-world applications require both prediction and optimization, and have been cast as PtO problems. For example, recommender systems need to predict user-item affinity to determine which titles to display (Wilder, Dilkina, and Tambe 2019), while portfolio optimization uses stock price predictions to construct high-performing portfolios (Bengio 1997). In the context of AI for Social Good, PtO has been used to plan intervention strategies by predicting how different subgroups will respond to interventions (Wang et al. 2022).

The central research question of PtO is, "How can we use the structure of an optimization problem to learn predictive models that perform better *for that specific decision-making*

task?". In this paper, we refer to the broad class of methods used to achieve this goal as *Decision-Focused Learning* (DFL). Recently, multiple papers have proposed learning task-specific loss functions for DFL (Chung et al. 2022; Lawless and Zhou 2022; Shah et al. 2022). The intuition for these methods can be summarized in terms of the Anna Karenina principle—while perfect predictions lead to perfect decisions, different kinds of imperfect predictions have different impacts on downstream decision-making. Such loss functions, then, attempt to use learnable parameters to capture how bad different kinds of prediction errors are for the decision-making task of interest. For example, a Mean Squared Error (MSE) loss may be augmented with tunable parameters to assign different weights to different true labels. Then, a model trained on such a loss is less likely to make the kinds of errors that affect the quality of downstream decisions.

Learning task-specific loss functions poses two major challenges. First, learning the relationship between predictions and decisions is challenging. To make learning this relationship more tractable, past approaches learn different loss functions for each instance of the decision-making task, each of which locally approximates the behavior of the optimization task. However, the inability to leverage training samples across different such instances can make learning loss functions sample inefficient, especially for approaches that require a large number of samples to learn. This is especially problematic because creating the dataset for learning these loss functions is the most expensive part of the overall approach. In this paper, rather than learning separate loss functions for each decision-making instance, we learn a mapping from the feature space of the predictive model to the parameters of different local loss functions. This 'feature-based parameterization' gives us the best of both worlds—we retain the simplicity of learning local loss functions while still being able to generalize across different decision-making instances. In addition to increasing efficiency, this reparameterization also ensures that the learned loss functions are *Fisher Consistent*—a fundamental theoretical property that ensures that, in the limit of infinite data and model capacity, optimizing for the loss function leads to optimal decision-making. Past methods for learning loss functions do not satisfy even this basic theoretical property!

The second challenge with learning loss functions is that it presents a chicken-and-egg problem—to obtain the distribu-

tion of predictions over which the learned loss function must accurately approximate the true decision quality, a predictive model is required, yet to obtain such a model, a loss function is needed to train it. To address this, Shah et al. (2022) use a simplification we call the "localness of prediction", i.e., they assume that predictions will be 'close' to the true labels, and generate candidate predictions by adding random noise to the true labels. However, this doesn't take into account the kinds of predictions that models actually generate and, as a result, can lead to the loss functions being optimized for unrealistic predictions. In contrast, Lawless and Zhou (2022) and Chung et al. (2022) use a predictive model trained using MSE to produce a single representative sample that is used to construct a simple loss function. However, this single sample is not sufficient to learn complex loss functions. We explicitly formulate the goal of sampling a *distribution* of realistic model predictions, and introduce a technique that we call *model-based sampling* to efficiently generate such samples. Because these interventions allow us to move away from localness-based simplifications, we call our loss functions 'Efficient Global Losses' or *EGLs* 🦅.

In addition to our theoretical Fisher Consistency results (Section 4.1), we show the merits of EGLs empirically by comparing them to past work on four domains (Section 6). First, we show that in one of our key domains, model-based sampling is essential for good performance, with EGLs outperforming even the best baseline by nearly 200%. The key characteristic of this domain is that it breaks the localness assumption (Section 5.1), which causes past methods to fail catastrophically. Second, we show that EGLs achieve state-of-the-art performance on the remaining three domains from the literature, *despite* having an order of magnitude fewer samples than comparable methods in two out of three of them. This improvement in sample efficiency translates to a reduction in the time taken to learn task-specific loss functions, resulting in an order-of-magnitude speed-up. All-in-all, we believe that these improvements bring DFL one step closer to being accessible in practice.

2 Background

2.1 Predict-then-Optimize

There are two steps in Predict-then-Optimize (PtO). In the *Predict* step, a learned predictive model $M_{\boldsymbol{\theta}}$ is used to make predictions about uncertain quantities $[\hat{y}_1, \ldots, \hat{y}_N] = [M_{\boldsymbol{\theta}}(x_1), \ldots, M_{\boldsymbol{\theta}}(x_N)]$ based on some features $[x_1, \ldots, x_N]$. Next, in the *Optimize* step, these predictions are aggregated as $\hat{\boldsymbol{y}} = [\hat{y}_1, \ldots, \hat{y}_N]$ and used to parameterize an optimization problem $\boldsymbol{z}^*(\hat{\boldsymbol{y}})$:

$$\boldsymbol{z}^*(\hat{\boldsymbol{y}}) = \arg\max_{\boldsymbol{z}} \ f(\boldsymbol{z}; \hat{\boldsymbol{y}}), \quad s.t. \ \boldsymbol{z} \in \boldsymbol{\Omega}$$

where f is the objective and $\boldsymbol{\Omega} \subseteq \mathbb{R}^{\dim(\boldsymbol{z})}$ is the feasible region. The solution to this optimization task $\boldsymbol{z} = \boldsymbol{z}^*(\hat{\boldsymbol{y}})$ provides the optimal decision for the predictions $\hat{\boldsymbol{y}}$. We call a full set of inputs $\hat{\boldsymbol{y}}$ or $\boldsymbol{y} = [y_1, \ldots, y_N]$ to the optimization problem an *instance* of the decision-making task.

However, the optimal decision $\boldsymbol{z}^*(\hat{\boldsymbol{y}})$ for the predictions $\hat{\boldsymbol{y}}$ may not be optimal for the true labels $\boldsymbol{y}$. To evaluate the

Decision Quality (DQ) of a set of predictions $\hat{\boldsymbol{y}}$, we measure how well the decisions they induce $\boldsymbol{z}^*(\hat{\boldsymbol{y}})$ perform on the set of *true labels* $\boldsymbol{y}$ with respect to the objective function f:

$$DQ(\hat{\boldsymbol{y}}, \boldsymbol{y}) = f(\boldsymbol{z}^*(\hat{\boldsymbol{y}}); \boldsymbol{y}) \qquad (1)$$

The central question of Predict-then-Optimize, then, is about how to learn predictive models $M_{\boldsymbol{\theta}}$ that have high DQ. When models are trained in a task-agnostic manner, e.g., to minimize Mean Squared Error (MSE), there can be a mismatch between predictive accuracy and DQ, and past work (see Section 3) has shown that the structure of the optimization problem $\boldsymbol{z}^*$ can be used to learn predictive models with better DQ. We refer to this broad class of methods for tailoring predictive models to decision-making tasks as Decision-Focused Learning (DFL) and describe one recent approach below.

2.2 Task-Specific Loss Functions

Multiple authors have suggested learning task-specific loss functions for DFL (Chung et al. 2022; Lawless and Zhou 2022; Shah et al. 2022). These approaches add learnable parameters to standard loss functions (e.g., MSE) and tune them, such that the resulting loss functions approximate the 'regret' in DQ for 'typical' predictions. Concretely, for the distribution over predictions $\hat{\boldsymbol{y}} = [M_{\boldsymbol{\theta}}(x_1), \ldots, M_{\boldsymbol{\theta}}(x_N)]$ of the model $M_{\boldsymbol{\theta}}$, the goal is to choose a loss function $L_{\boldsymbol{\phi}}$ with parameters $\boldsymbol{\phi}$ such that:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{x}, \boldsymbol{y}, \hat{\boldsymbol{y}}} \left[\left(L_{\boldsymbol{\phi}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) - DQ_{\text{regret}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) \right)^2 \right]$$

$$\text{where} \quad DQ_{\text{regret}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) \equiv DQ(\boldsymbol{y}, \boldsymbol{y}) - DQ(\hat{\boldsymbol{y}}, \boldsymbol{y}) \quad (2)$$

where DQ is defined in Equation (1). Note here that the first term in DQ_{regret} is a constant w.r.t. $\hat{y}$, so minimizing DQ_{regret} is equivalent to maximizing DQ. Adding the $DQ(\boldsymbol{y}, \boldsymbol{y})$ term, however, makes DQ_{regret} behave more like a loss function—a minimization objective with a minimum value of 0 at $\hat{\boldsymbol{y}} = \boldsymbol{y}$. As a result, parameterized versions of simple loss functions can learn the structure of DQ_{regret} (and thus DQ).

A meta-algorithm for learning predictive models $M_{\boldsymbol{\theta}}$ using task-specific loss functions is as follows:

1. **Sampling $\tilde{\boldsymbol{y}}$:** Generate K candidate predictions $\tilde{\boldsymbol{y}}^k = [y_1 \pm \epsilon_1, \ldots, y_N \pm \epsilon_N]$, e.g., by adding Gaussian noise $\epsilon_n \sim \mathcal{N}(0, \sigma)$ to each true label y_n. This strategy is motivated by the 'localness of predictions' assumption, i.e., that predictions will be close to the true labels.

2. **Generating dataset:** Run a optimization solver on the sampled predictions $\tilde{\boldsymbol{y}}$ to get the corresponding decision quality values $DQ_{\text{regret}}(\tilde{\boldsymbol{y}}, \boldsymbol{y})$. This results in a dataset of the form $[(\tilde{\boldsymbol{y}}^1, DQ_{\text{regret}}^1), \ldots, (\tilde{\boldsymbol{y}}^K, DQ_{\text{regret}}^K)]$ for each instance $\boldsymbol{y}$ in the training and validation set.

3. **Learning Loss Function(s):** Learn loss function(s) that minimize the MSE on the dataset from Step 2. Lawless and Zhou (2022) and Chung et al. (2022) re-weight the MSE loss *for each instance*:

$$\text{L\&Z}_w^{\boldsymbol{y}}(\hat{\boldsymbol{y}}) = w^{\boldsymbol{y}} \cdot ||\hat{\boldsymbol{y}} - \boldsymbol{y}||_2^2 \qquad (3)$$

Shah et al. (2022) propose 2 families of loss functions, which they call 'Locally Optimized Decision Losses'

(LODLs). The first adds learnable weights to MSE *for each prediction* that comprises the instance $\hat{\boldsymbol{y}}$. The second is more general—an arbitrary quadratic function of the predictions that comprise $\hat{\boldsymbol{y}}$, where the learned parameters are the coefficients of each polynomial term.

$$\text{WMSE}_{\boldsymbol{w}}^{\boldsymbol{y}}(\hat{\boldsymbol{y}}) = \sum_{n=1}^{N} w_n^{\boldsymbol{y}} \cdot (\hat{y}_n - y_n)^2$$

$$\text{Quadratic}_H^{\boldsymbol{y}}(\hat{\boldsymbol{y}}) = (\hat{\boldsymbol{y}} - \boldsymbol{y})^T H^{\boldsymbol{y}}(\hat{\boldsymbol{y}} - \boldsymbol{y}) \qquad (4)$$

The parameters for these losses $w > w_{\min} > 0$ and $H = L^T L + w_{\min} \cdot I$ are constrained to ensure that the learned loss is convex. They also propose 'Directed' variants of each loss in which the parameters to be learned are different based on whether $(\hat{y} - y) > 0$ or not. These parameters are then learned for every instance $\boldsymbol{y}$, e.g., using gradient descent.

4. **Learning predictive model $M_{\boldsymbol{\theta}}$:** Train the predictive model $M_{\boldsymbol{\theta}}$ on the loss functions learned in the previous step, e.g., a random forest (Chung et al. 2022), a neural network (Shah et al. 2022), or a linear model (Lawless and Zhou 2022).

In this paper, we propose two modifications to the meta-algorithm above. We modify Step 1 in Section 5 and Step 3 in Section 4. Given that these contributions help overcome the challenges associated with the losses being "local", we call our new method *EGL* 🦋 (Efficient Global Losses).

3 Related Work

In Section 2.2, we contextualize why the task-specific loss function approaches of Chung et al. (2022); Lawless and Zhou (2022); Shah et al. (2022) make sense, and use the intuition to scale model-based sampling and better empirically evaluate the utility of different aspects of learned losses.

In addition to learning loss functions, there are alternative approaches to DFL. The most common of these involves optimizing for the decision quality directly by modeling the Predict-then-Optimize task end-to-end and differentiating through the optimization problem (Agrawal et al. 2019; Amos et al. 2018; Donti, Amos, and Kolter 2017). However, discrete optimization problems do not have informative gradients, and so cannot be used as-is in an end-to-end pipeline. To address this issue, most past work either constructs task-dependent surrogate problems that *do* have informative gradients for learning predictive models (Ferber et al. 2020; Mensch and Blondel 2018; Tschiatschek, Sahin, and Krause 2018; Wilder, Dilkina, and Tambe 2019; Wilder et al. 2019) or propose loss functions for specific classes of optimization problems (e.g. with linear objectives (Elmachtoub and Grigas 2021; Mulamba et al. 2021)). However, the difficulty of finding good task-specific relaxations for arbitrary optimization problems limits the adoption of these techniques in practice. On the other hand, learning a loss function is more widely applicable as it applies to *any optimization problem* and only requires access to a solver for the optimization problem; as a result, we focus on improving this approach in this paper.

4 Part One: Feature-based Parameterization

The challenge of learning the $DQ_{\text{regret}}(\hat{\boldsymbol{y}}, \boldsymbol{y})$ function is that it is as hard as learning a closed-form solution to an arbitrary optimization problem. To make the learning problem easier, past approaches typically learn multiple local approximations, e.g., $DQ_{\text{regret}}^i(\hat{\boldsymbol{y}}_i)$ *for every decision-making instance $\boldsymbol{y}_i$* in the training and validation set. This simplification trades off the complexity of learning a single complex DQ_{regret} for the cost of learning many simpler DQ_{regret}^i functions:

$$\underbrace{DQ_{\text{regret}}(\hat{\boldsymbol{y}}, \boldsymbol{y})}_{\text{complex}} \approx \underbrace{DQ_{\text{regret}}^i(\hat{\boldsymbol{y}}_i)}_{\text{simple}}, \ \forall \hat{\boldsymbol{y}}_i \approx \boldsymbol{y}_i + \epsilon, \ \forall i \in [N]$$

However, this is problematic because calculating DQ_{regret} for each of the sampled predictions $\tilde{\boldsymbol{y}}$ is the most expensive step in learning task-focused loss functions (see Section 6.2). Specifically, learning each DQ_{regret}^i can require as many as $\Omega(\dim(\boldsymbol{y})^2)$ samples (for the 'DirectedQuadratic' loss function from Shah et al. (2022)), leading to the need for $N \cdot \dim(\boldsymbol{y})^2$ samples overall. As a result, this approach does not scale as the number of instances ($i \in [N]$) in the dataset or the size of the optimization problem ($\dim(\boldsymbol{y})$) increases.

To make learning loss functions more scalable, we have to find a way to trade-off the hardness of learning a global approximation against the cost of learning a local approximation. The standard machine learning approach for this is to replace learning DQ_{regret}^i with a loss L that allows pooling samples across different instances $i \in [N]$ and predictions $\dim(\boldsymbol{y})$. However, in doing this, two important design choices have to be made—(1) What should be the functional form of L?, and (2) What should the inputs to L be?

For the first, we could set L to simply be some neural network model and try to learn some sort of global approximation to DQ_{regret}. However, in addition to the fact that this does not address the "hardness" of learning a global approximation, Shah et al. (2022) show that the lack of convexity of the learned neural networks can sometimes lead to catastrophic failures, i.e., models that perform worse than random guessing. Building on this insight, we propose using a neural network to instead learn a mapping to the *parameters of a convex-by-construction loss function family*. That way, regardless of the learned parameters, we ensure that $\hat{\boldsymbol{y}}^* = \boldsymbol{y}$ is a minimizer of the resulting loss function and so the learned loss is guaranteed to never fail catastrophically.

For the second, we propose using the features x_i associated with the prediction y_i as the input to our learned loss. To ensure that $\hat{\boldsymbol{y}}^* = \boldsymbol{y}$ regardless of how well we model DQ_{regret}, we show in Section 4.1 that the loss functions must have the same set of parameters for a given set of features $\boldsymbol{x}$ in order to be Fisher Consistent (discussed in more detail below).

Putting both of these together, EGLs 🦋 learn a mapping $P_{\boldsymbol{\psi}}(x)$ from the features x of any prediction in the dataset to the corresponding parameters in convex-by-construction LODL loss families as follows:

- **WeightedMSE:** We learn a mapping $P_{\boldsymbol{\psi}} : x \to w$ from the features x of a optimization parameter y to its associated 'weight' w. Intuitively, the weight w encodes how important a given prediction is, and so EGLs learn to predict the impact of different predictions' errors on DQ_{regret}.
- **Quadratic:** For every pair of predictions $\hat{y}_i$ and $\hat{y}_j$, we learn a mapping $P_{\boldsymbol{\psi}} : (x_i, x_j) \to L_{ij}$ where $L = [[L_{ij}]]$

is the matrix that parameterizes the loss function (see Equation (4)).

- **Directed Variants:** Instead of learning a mapping from the features x to a single parameter, we instead learn a mapping from $x \to [w^+, w^-]$ for 'Directed WeightedMSE' and $(x_i, x_j) \to [L^{++}, L^{\pm}, L^{-+}, L^{--}]$ for 'Directed Quadratic'.

We then optimize for the optimal parameters ψ^* of the learned losses along the lines of past work:

$$\psi^* = \arg\min_{\psi} \mathbb{E}_{\boldsymbol{x}, \boldsymbol{y}, \hat{\boldsymbol{y}}} \left[\left(L_{P_\psi(\boldsymbol{x})}(\hat{\boldsymbol{y}}, \boldsymbol{y}) - DQ_{\text{regret}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) \right)^2 \right]$$

where $L_\phi = L_{P_\psi(\boldsymbol{x})}$ is the learned loss function. For our experiments, the model P_ψ is a 4-layer feedforward neural network with a hidden dimension of 500 trained using gradient descent. Given that we learn a mapping from the features of a given prediction x to the corresponding loss function parameter(s), we call our resulting approach *feature-based parameterization (FBP)*.

4.1 Fisher Consistency

One desirable property of a Predict-then-Optimize loss function is that, in the limit of infinite data and model capacity, the optimal prediction induced by the loss also minimizes the decision quality regret. If this is true, we say that loss ℓ is "Fisher Consistent" w.r.t. the decision quality regret DQ_{regret} (Elmachtoub and Grigas 2021).

Definition 4.1 (Fisher Consistency). A loss $\ell(\hat{\boldsymbol{y}}, \boldsymbol{y})$ is said to be Fisher Consistent with respect to the decision quality regret DQ_{regret} if the set of minimizers of the loss function $\hat{\boldsymbol{y}}^*(\boldsymbol{x}) = \arg\min_{\hat{\boldsymbol{y}}} \mathbb{E}_{\boldsymbol{y}|\boldsymbol{x}}[\ell(\hat{\boldsymbol{y}}, \boldsymbol{y})]$ also minimize DQ_{regret} for all possible distributions $P(\boldsymbol{x}, \boldsymbol{y})$.

However, the methods proposed in past work do not satisfy this property even for the simplest of cases—in which the objective f of the optimization problem $\boldsymbol{z}^*$ is linear. Concretely:

Theorem 4.2. *Weighting-the-MSE losses are not Fisher Consistent for Predict-then-Optimize problems in which the optimization function $\boldsymbol{z}^*$ has a linear objective.*

Proof. We show a proof by counter-example below. Consider a PtO problem in which the goal is to (a) predict the utility of a resource for two individuals (say, A and B), and then (b) give the resource to the individual with higher utility. Consider the utilities to be drawn from:

$$\boldsymbol{y} = (y_A, y_B) = \begin{cases} (0, 0.55), & \text{with probability } 0.5 \\ (1, 0.55), & \text{with probability } 0.5 \end{cases}$$

The optimal decision, then, is to give the resource to individual B because $\mathbb{E}[\hat{y}_B] = 0.55 > \mathbb{E}[\hat{y}_A] = 0.5$.

To learn a predictive model using a Weighting-the-MSE loss, we follow the meta-algorithm in Section 2.2:

1. We sample K = 25 points in the "neighborhood" of each of the true labels $(0, 0.55)$ and $(1, 0.55)$. For simplicity, we add uniform-random noise $\epsilon^k \sim U[-1, 1]$ only to y_A to get $\tilde{\boldsymbol{y}}^k = (y_A \pm \epsilon^k, y_B)$.
2. We calculate DQ_{regret} for each sample. We plot DQ_{regret} vs $\hat{y}_A$ in Figure 1 (given $\hat{y}_B$ is fixed).
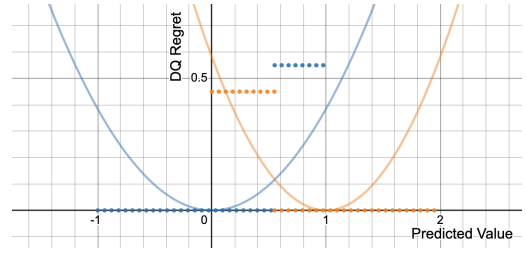

Figure 1: **A plot of DQ_{regret} vs. $\hat{y}_A$** (for fixed $\hat{y}_B$). The samples from Step 1 correspond to the blue dots for the $(0, 0.55)$ instance and the orange dots for the $(1, 0.55)$ instance. We also plot the learned weighted MSE loss for each instance using solid lines in their corresponding colors.

3. Based on this dataset, we fit a loss of the form given by Equation (3). Because there is only one weight being learned here, this can be seen as either a WeightedMSE LODL from Shah et al. (2022) or a reweighted task-loss from Lawless and Zhou (2022), as seen in Figure 1.
4. Finally, we estimate a predictive model based on this loss. The optimal prediction given this loss is $\hat{\boldsymbol{y}}^* = (\hat{y}_A^*, \hat{y}_B^*) \approx (0.602, 0.55)$ (see Appendix B for details).

Under this predictive model, the optimal decision would be to give the resource to individual A because they have higher predicted utility, i.e., $\hat{y}_A^* > \hat{y}_B^*$. *But this is suboptimal!* □

More generally, because "WeightedMSE" can be seen as a special case of the other methods in Shah et al. (2022), none of the loss functions in past work are Fisher Consistent! To gain intuition for why this happens, let us analyze the predictions that are induced by weighting-the-MSE type losses:

Lemma 4.3. *The optimal prediction $\hat{y}^*(x)$ for some feature x given a weighted MSE loss function with weights $w^{\boldsymbol{y}}$ associated with the label $y \in \boldsymbol{y}$ is $\hat{y}^*(x) = \frac{\mathbb{E}_{\boldsymbol{y}|x}[w^{\boldsymbol{y}} \cdot y]}{\mathbb{E}_{\boldsymbol{y}|x}[w^{\boldsymbol{y}}]}$, given infinite model capacity.*

The proof is presented in Appendix A. In their paper, Elmachtoub and Grigas (2021) show that the optimal prediction that minimizes DQ_{regret} is $\hat{y}^*(x) = \mathbb{E}_{\boldsymbol{y}|x}[y]$, i.e., the optimal prediction should depend only on the value of the labels corresponding to that feature. However, $\hat{y}^*(x) = \frac{\mathbb{E}_{\boldsymbol{y}|x}[w^{\boldsymbol{y}} \cdot y]}{\mathbb{E}_{\boldsymbol{y}|x}[w^{\boldsymbol{y}}]}$ in weighted MSE losses, and so the optimal prediction depends on not only the labels but also the *weight* associated with them. While this is not a problem by itself, the weight learned for a label $y \in \boldsymbol{y}$ is dependent on not only the label itself but also the *other labels $\boldsymbol{y}_{-y}$ in that instance*. In the context of the counter-example in the proof of Theorem 4.2, the weight associated with individual A is dependent on the utility of individual B (via DQ_{regret}). As a result, it is possible to create a distribution $P(\boldsymbol{x}, \boldsymbol{y})$ for which such losses will not be Fisher Consistent. However, this is not true for WeightedMSE with FBP!

Theorem 4.4. *WeightedMSE with FBP is Fisher Consistent for Predict-then-Optimize problems in which the optimization function $\boldsymbol{z}^*$ has a linear objective.*

Proof. In WeightedMSE with FBP, the weights associated with some feature x are not independently learned for each instance $\boldsymbol{y}$ but are instead a *function of the features* x. As a result, the weight $w^{\boldsymbol{y}}$ associated with that feature is the same across all instances, i.e., $w^{\boldsymbol{y}} = w(x), \forall \boldsymbol{y}$. Plugging that into the equation from Lemma 4.3:

$$\hat{y}^*(x) = \frac{\mathbb{E}_{\boldsymbol{y}|x}[w^{\boldsymbol{y}} \cdot y]}{\mathbb{E}_{\boldsymbol{y}|x}[w^{\boldsymbol{y}}]} = \frac{w(x) \cdot \mathbb{E}_{\boldsymbol{y}|x}[y]}{w(x)} = \mathbb{E}_{\boldsymbol{y}|x}[y]$$

which is a minimizer of DQ_{regret} (Elmachtoub and Grigas 2021). Thus, WeightedMSE with FBP is Fisher Consistent. $\square$

5   Part Two: Model-based Sampling

Loss functions serve to give feedback to the model. However, to make them easier to learn, their expressivity is often limited. As a result, they cannot estimate DQ_{regret} accurately for *all* possible predictions but must instead concentrate on a subset of "realistic predictions" for which the predictive model $M_{\boldsymbol{\theta}}$ will require feedback during training. However, there is a chicken-and-egg problem in learning loss functions on realistic predictions—a model is needed to generate such predictions, but creating such a model would in turn require its own loss function to train on.

Past work has made the assumption that the predictions $\hat{\boldsymbol{y}}$ will be close to the actual labels $\boldsymbol{y}$ to efficiently generate potential predictions $\tilde{\boldsymbol{y}}$. However, if the assumption does not hold, Gaussian sampling may not yield good results, as seen in Section 5.1. Instead, in this paper, we propose an alternative: *model-based sampling (MBS)*. Here, to generate a *distribution* of potential predictions using this approach, we train a predictive model $M_{\boldsymbol{\theta}}$ on a standard loss function (e.g., MSE). Then, at equally spaced intervals during the training process, we use the intermediate model to generate predictions for each problem instance in the dataset. These form the set of *potential predictions* $\tilde{\boldsymbol{y}}$ based on which we create the dataset and learn loss functions. The hyperparameters associated with this approach are:

- **Number of Models:** Instead of sampling predictions from just one model, we can train multiple models to increase the diversity of the generated predictions. In our experiments, we choose from $\{1, 5, 10\}$ predictive models.
- **LR** and **Number of Training Steps:** The learning rates are chosen from $\{10^{-6}, 10^{-5}, \dots, 1\}$ with a possible cyclic schedule (Smith 2017). We use a maximum of 50000 updates across all the models.

Empirically, we find that a high learning rate and large number of models works best. Both of these choices increase the diversity of the generated samples and help create a richer dataset for learning loss functions.

5.1   Localness of Predictions

To illustrate the utility of model-based sampling, we analyze the Cubic Top-K domain proposed by Shah et al. (2022). The goal in this domain is to fit a linear model to approximate a more complex cubic relationship between x and y (Figure 2, Appendix C). This could be motivated by explainability (Rudin 2019; Hughes et al. 2018), data efficiency, or

simplicity. The localness assumption breaks here because it is not possible for linear models (or low-capacity models more generally) to closely approximate the true labels of a more complex data-generating process. The objective of the learned loss function, then, is to provide information about *what kind of suboptimal predictions are better than others*.

Learned losses accomplish this by first generating plausible predictions and then learning how different sorts of errors change the decision quality. In this domain, the decision quality is solely determined by the point with the highest predicted utility. As can be seen in Figure 2 (Appendix C), the highest values given $x \sim U[-1, 1]$ are either at $x = -0.5$ or $x = 1$. In fact, because the function is flatter around $x = -0.5$, there are more likely to be large values there. When Gaussian sampling (Shah et al. 2022) is used to generate candidate predictions, the highest sampled values are also more likely to be at $x = -0.5$ because the added noise has a mean of zero. However, *a linear model cannot have a maximum value at $x = -0.5$, only $x \in \{-1, 1\}$*. As a result, the loss functions learned based on the samples from this method focus on the wrong subset of labels and lead to poor downstream performance. On the other hand, the candidate predictions generated by model-based sampling are the outputs of linear models, allowing the loss functions to take this into account. We visualize this phenomenon in Figure 3.

In their paper, Shah et al. (2022) propose a set of "directed models" to make LODLs perform well in this domain. However, these models only learn useful loss functions because the value of the label at $x = 1$ is *slightly higher* than the value at $x = -0.5$. To show this, we create a variant of this domain called "(Hard) Cubic Top-K" in which $y_{x=-0.5} > y_{x=1}$. Then, in Table 1, we see that even the "directed" LODLs fail catastrophically in this domain, while the loss functions learned with model-based sampling perform well.

6   Experiments

In this section, we validate EGLs empirically on four domains from the literature. We provide brief descriptions of the domains below but refer the reader to the corresponding papers for more details.

Cubic Top-K (Shah et al. 2022) Learn a model whose top-k predictions have high corresponding true labels.

- *Predict:* Predict resource n's utility $\hat{y}_n$ using a linear model with feature $x_n \sim U[-1, 1]$. The true utility is $y_n = 10x_n^3 - 6.5x_n$ for the standard version of the domain and $y_n = 10x_n^3 - \boldsymbol{7.5}x_n$ for the 'hard' version. The predictive model is linear, i.e., $M_{\boldsymbol{\theta}}(x) = mx + c$.
- *Optimize:* Out of $N = 50$ resources, choose the top $K = 1$ resources with highest utility.

Web Advertising (Wilder, Dilkina, and Tambe 2019) The objective of this domain is to predict the Click-Through-Rates (CTRs) of different (user, website) pairs such that good websites to advertise on are chosen.

- *Predict:* Predict the CTRs $\hat{\boldsymbol{y}}_m$ for $N = 10$ fixed users on $M = 5$ websites using the website's features x_m. The features for each website are obtained by multiplying the true CTRs $\boldsymbol{y}_m$ from the Yahoo! Webscope Dataset (Yahoo! 2007) with a random $N \times N$ matrix A, resulting in $\boldsymbol{x}_m =$

Table 1: **Overall Results.** The entries represent the Mean Normalized Test DQ $\pm$ SEM. Methods that are not applicable to specific domains are denoted by '-'. Bolded values represent the set of methods that outperform the others by a statistically significant margin (p-value < 0.05). Our method achieves state-of-the-art performance without any handcrafting and an order of magnitude fewer samples.

Category	Method	New Domain	Domains from the Literature		
		(Hard) Cubic Top-K	Cubic Top-K	Web Advertising	Portfolio Optimization
2-Stage	MSE	-0.65 $\pm$ 0.04	-0.50 $\pm$ 0.06	0.60 $\pm$ 0.04	0.04 $\pm$ 0.00
Expert-crafted Surrogates	SPO+	-0.68 $\pm$ 0.00	**0.96 $\pm$ 0.00**	-	-
	Entropy-Regularized Top-K	0.24 $\pm$ 0.08	**0.96 $\pm$ 0.00**	-	-
	Multilinear Relaxation	-	-	0.74 $\pm$ 0.01	-
	Differentiable QP	-	-	-	0.141 $\pm$ 0.003
Learned Losses	L&Z (1 Sample)	-0.68 $\pm$ 0.00	-0.96 $\pm$ 0.00	0.65 $\pm$ 0.02	0.133 $\pm$ 0.005
	LODL (32 Samples)	-0.68 $\pm$ 0.00	-0.38 $\pm$ 0.29	0.84 $\pm$ 0.04	**0.146 $\pm$ 0.003**
	LODL (2048 Samples)	-0.67 $\pm$ 0.01	**0.96 $\pm$ 0.00**	**0.93 $\pm$ 0.01**	**0.154 $\pm$ 0.005**
	EGL 🦅 (32 Samples)	**0.69 $\pm$ 0.00**	**0.96 $\pm$ 0.00**	**0.95 $\pm$ 0.01**	**0.153 $\pm$ 0.004**

$A\boldsymbol{y}_m$. The predictive model $M_{\boldsymbol{\theta}}$ is a 2-layer feedforward network with a hidden dimension of 500.

- *Optimize:* Choose which $K = 2$ websites to advertise on such that the expected number of users who click on an advertisement at least once (according to the CTR matrix) is maximized. The objective to be maximized is $\boldsymbol{z}^*(\hat{\boldsymbol{y}}) = \arg\max_{\boldsymbol{z}} \sum_{j=0}^{N}(1 - \prod_{i=0}^{M}(1 - z_i \cdot \hat{y}_{ij}))$, where z_i can be either 0 or 1. This is a submodular maximization problem.

Portfolio Optimization (Donti, Amos, and Kolter 2017) Based on the Markovitz model (Markowitz and Todd 2000), the aim is to predict future stock prices in order to create a portfolio that has high returns but low risk.

- *Predict:* Predict the future stock price y_n for each stock n using its historical data x_n. The historical data includes information on 50 stocks obtained from the QuandlWIKI dataset (Quandl 2022). The model $M_{\boldsymbol{\theta}}$ is a 2-layer feedforward network with a hidden dimension of 500.
- *Optimize:* Choose a distribution $\boldsymbol{z}$ over stocks to maximize $\boldsymbol{z}^T\boldsymbol{y} - \lambda \cdot \hat{\boldsymbol{y}}^T Q\hat{\boldsymbol{y}}$ based on a known correlation matrix Q of stock prices. Here, $\lambda = 0.001$ represents the constant for risk aversion.

For each set of experiments, we run 10 experiments with different train-test splits, and randomized initializations of the predictive model and loss function parameters. Details of the computational resources and hyperparameter optimization used are given in Appendix D.

For all of these domains, the metric of interest is the decision quality achieved by the predictive model $M_{\boldsymbol{\theta}}$ on the hold-out test set when trained with the loss function in question. However, given that the scales of the decision quality for each domain vary widely, we linearly re-scale the value such that 0 corresponds to the DQ of making predictions uniformly at random $\hat{y} = \epsilon \sim U[0, 1]$ and 1 corresponds to making perfect predictions $\hat{y} = y$. Concretely:

$$\text{Normalized DQ}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{DQ(\hat{\boldsymbol{y}}, \boldsymbol{y}) - DQ(\epsilon, \boldsymbol{y})}{DQ(\boldsymbol{y}, \boldsymbol{y}) - DQ(\epsilon, \boldsymbol{y})}$$

6.1 Overall Results

We compare our approach against the following baselines from the literature in Table 1:

- **MSE:** A standard regression loss.
- **Expert-crafted Surrogates:** The end-to-end approaches described in Section 3 that require handcrafting differentiable surrogate optimization problems for each domain separately (Donti, Amos, and Kolter 2017; Wang et al. 2020; Elmachtoub and Grigas 2021; Xie et al. 2020; Wilder et al. 2019).
- **L&Z:** Lawless and Zhou (2022)'s approach for learning losses (equivalent to Chung et al. (2022)).
- **LODL:** Shah et al. (2022)'s approach for learning loss functions. Trained using 32 and 2048 samples.

(Hard) Cubic Top-K We empirically verify our analysis from Section 5.1 by testing different baselines on our proposed "hard" top-k domain. In Table 1, we see that all our baselines perform extremely poorly in this domain. Even the expert-crafted surrogate only achieves a DQ of 0.24 while EGLs achieve the best possible DQ of 0.69; this corresponds to a gain of nearly 200% for EGLs.

Domains for the Literature We find that our method reaches state-of-the-art performance in all the domains from the literature. In fact, we see that EGLs achieve similar performance to LODLs with an order of magnitude fewer samples in two out of three domains. In Section 6.2 below, we see that this corresponds to *an order of magnitude speed-up over learning LODLs of similar quality!*

6.2 Computational Complexity Experiments

We saw in Section 6.1 that EGLs perform as well as LODLs with an order of magnitude fewer samples. In Table 3, we show how this increased sample efficiency translates to differences in runtime. We see that, by far, most of the time in learning LODLs is spent in step 2 of our meta-algorithm. As a result, despite the fact that EGLs take longer to perform

Table 2: **Comparison to LODLs.** MBS → Model-based Sampling, FBP → Feature-based Parameterization, and EGL = LODL + MBS + FBP. The entries represent the Mean Normalized Test DQ ± SEM. EGLs improve the DQ for almost every choice of loss function family and domain.

Domain	Method	Normalized Test Decision Quality			
		Directed Quadratic	Directed WeightedMSE	Quadratic	WeightedMSE
Cubic Top-K	LODL	-0.38 ± 0.29	-0.86 ± 0.10	-0.76 ± 0.19	-0.95 ± 0.01
	LODL (2048 samples)	-0.94 ± 0.01	0.96 ± 0.00	-0.95 ± 0.01	-0.96 ± 0.00
	EGL (MBS)	$\mathbf{0.96 \pm 0.00}$	$\mathbf{0.96 \pm 0.00}$	0.77 ± 0.13	$\mathbf{0.77 \pm 0.13}$
	EGL (FBP)	0.58 ± 0.26	$\mathbf{0.96 \pm 0.00}$	-0.28 ± 0.21	-0.77 ± 0.11
	EGL (Both)	$\mathbf{0.96 \pm 0.00}$	0.77 ± 0.13	$\mathbf{0.96 \pm 0.00}$	$\mathbf{0.77 \pm 0.13}$
Web Advertising	LODL	0.75 ± 0.05	0.72 ± 0.03	0.84 ± 0.04	0.71 ± 0.03
	LODL (2048 samples)	0.93 ± 0.01	0.84 ± 0.02	0.93 ± 0.01	0.78 ± 0.03
	EGL (MBS)	0.86 ± 0.03	$\mathbf{0.83 \pm 0.03}$	0.78 ± 0.06	0.77 ± 0.04
	EGL (FBP)	0.93 ± 0.02	0.80 ± 0.03	$\mathbf{0.92 \pm 0.01}$	0.75 ± 0.04
	EGL (Both)	$\mathbf{0.95 \pm 0.01}$	0.78 ± 0.06	$\mathbf{0.92 \pm 0.02}$	$\mathbf{0.81 \pm 0.04}$
Portfolio Optimization	LODL	$\mathbf{0.146 \pm 0.003}$	0.136 ± 0.003	0.145 ± 0.003	0.122 ± 0.003
	LODL (2048 samples)	0.154 ± 0.005	0.141 ± 0.004	0.147 ± 0.004	0.113 ± 0.014
	EGL (MBS)	0.135 ± 0.011	0.138 ± 0.010	0.146 ± 0.015	0.108 ± 0.009
	EGL (FBP)	0.139 ± 0.005	0.141 ± 0.008	$\mathbf{0.147 \pm 0.008}$	0.136 ± 0.004
	EGL (Both)	0.134 ± 0.013	$\mathbf{0.127 \pm 0.011}$	0.145 ± 0.011	$\mathbf{0.153 \pm 0.004}$

Table 3: **Time taken to run the meta-algorithm** (Section 2.2) for comparable WeightedMSE EGLs and LODLs on the Web Advertising domain. EGLs take only 6% of the LODLs' time to train.

Time Taken (in seconds)	Method	
	LODL	EGL 🦅
Samping $\tilde{y}$ (Step 1)	0.18 ± 0.01 (Gaussian Sampling)	0.48 ± 0.04 (MBS)
Generating Dataset (Step 2)	10376.65 ± 119.81 (2048 samples)	200.43 ± 4.24 (32 samples)
Learning Losses (Step 3)	53.67 ± 1.84 (Separate Losses)	445.67 ± 62.42 (FBP)
Total	10430.50 ± 131.66	646.58 ± 66.70

steps 1 and 3, the increase in sample efficiency results in an *order-of-magnitude* speedup over LODLs.

6.3 Ablation Study

In this section, we compare EGLs to their strongest competitor from the literature, i.e., LODLs (Shah et al. 2022). Specifically, we look at the low-sample regime—when 32 samples per instance are used to train both losses—and present our results in Table 2. We see that EGLs improve the decision quality for almost every choice of loss function family and domain. We further analyze Table 2 below.

Feature-based Parameterization (FBP): Given that this is the low-sample regime, 'LODL + FBP' almost always does better than just LODL. These gains are especially apparent in cases where adding more samples would improve LODL performance—the "Directed" variants in the Cubic Top-K

domain, and the "Quadratic" methods in the Web Advertising domain.

Model-based Sampling (MBS): This contribution is most useful in the Cubic Top-K domain, where the localness assumption is broken. Interestingly, however, MBS also improves performance in the other two domains where the localness assumption does not seem to be broken (Table 4 in Appendix D.1). We hypothesize that MBS helps here in two different ways:

1. *Increasing effective sample efficiency:* We see that, in the cases where FBP helps most, the gains from MBS stack with FBP. This suggests that MBS helps improve sample-efficiency. Our hypothesis is that model-based sampling allows us to focus on predictions that would lead to a "fork in the training trajectory", leading to improved performance with fewer samples.

2. *Helping WeightedMSE models:* MBS also helps improve the *worst-performing* WeightedMSE models in these domains which, when combined with FBP, outperform even LODLs with 2048 samples. This suggests that MBS does more than just increase sample efficiency. We hypothesize that MBS also reduces the search space by limiting the set of samples $\tilde{y}$ to "realistic predictions", allowing even WeightedMSE models that have fewer parameters to perform well in practice.

Portfolio Optimization: The results for this domain don't follow the trends noted above because there is a distribution shift between the validation and test sets in this domain (as the train/test/validation split is temporal instead of i.i.d.). In Table 5 (Appendix D.2), we see that EGLs outperform LODLs and follow the trends noted above if we measure their performance on the validation set, which is closer in time to training (and hence has less distribution shift).

References

Agrawal, A.; Amos, B.; Barratt, S.; Boyd, S.; Diamond, S.; and Kolter, J. Z. 2019. Differentiable convex optimization layers. *Advances in Neural Information Processing Systems*, 32.

Amos, B.; Jimenez, I.; Sacks, J.; Boots, B.; and Kolter, J. Z. 2018. Differentiable MPC for End-to-end Planning and Control. In *Advances in Neural Information Processing Systems*, volume 31.

Bengio, Y. 1997. Using a financial training criterion rather than a prediction criterion. *International journal of neural systems*, 8(04): 433–443.

Center, O. S. 1987. Ohio Supercomputer Center.

Chung, T.-H.; Rostami, V.; Bastani, H.; and Bastani, O. 2022. Decision-Aware Learning for Optimizing Health Supply Chains. *arXiv preprint arXiv:2211.08507*.

Donti, P.; Amos, B.; and Kolter, J. Z. 2017. Task-based end-to-end model learning in stochastic optimization. *Advances in Neural Information Processing Systems*, 30.

Elmachtoub, A. N.; and Grigas, P. 2021. Smart "predict, then optimize". *Management Science*.

Ferber, A.; Wilder, B.; Dilkina, B.; and Tambe, M. 2020. MIPaaL: Mixed integer program as a layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 1504–1511.

Hughes, M.; Hope, G.; Weiner, L.; McCoy, T.; Perlis, R.; Sudderth, E.; and Doshi-Velez, F. 2018. Semi-Supervised Prediction-Constrained Topic Models. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, 1067–1076.

Lawless, C.; and Zhou, A. 2022. A Note on Task-Aware Loss via Reweighing Prediction Loss by Decision-Regret. *arXiv preprint arXiv:2211.05116*.

Markowitz, H. M.; and Todd, G. P. 2000. *Mean-variance analysis in portfolio choice and capital markets*. John Wiley & Sons.

Mensch, A.; and Blondel, M. 2018. Differentiable dynamic programming for structured prediction and attention. In *International Conference on Machine Learning*, 3462–3471. PMLR.

Mulamba, M.; Mandi, J.; Diligenti, M.; Lombardi, M.; Bucarey, V.; and Guns, T. 2021. Contrastive Losses and Solution Caching for Predict-and-Optimize. In *Proceedings of the International Joint Conferences on Artificial Intelligence*.

Quandl. 2022. WIKI Various End-Of-Day Data.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5): 206–215.

Shah, S.; Wang, K.; Wilder, B.; Perrault, A.; and Tambe, M. 2022. Decision-Focused Learning without Decision-Making: Learning Locally Optimized Decision Losses. In *Advances in Neural Information Processing Systems*.

Smith, L. N. 2017. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, 464–472. IEEE.

Tschiatschek, S.; Sahin, A.; and Krause, A. 2018. Differentiable submodular maximization. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2731–2738.

Wang, K.; Shah, S.; Chen, H.; Perrault, A.; Doshi-Velez, F.; and Tambe, M. 2021. Learning MDPs from Features: Predict-Then-Optimize for Sequential Decision Making by Reinforcement Learning. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 8795–8806. Curran Associates, Inc.

Wang, K.; Verma, S.; Mate, A.; Shah, S.; Taneja, A.; Madhiwalla, N.; Hegde, A.; and Tambe, M. 2022. Decision-Focused Learning in Restless Multi-Armed Bandits with Application to Maternal and Child Care Domain. *arXiv preprint arXiv:2202.00916*.

Wang, K.; Wilder, B.; Perrault, A.; and Tambe, M. 2020. Automatically learning compact quality-aware surrogates for optimization problems. *Advances in Neural Information Processing Systems*, 33: 9586–9596.

Wilder, B.; Dilkina, B.; and Tambe, M. 2019. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 1658–1665.

Wilder, B.; Ewing, E.; Dilkina, B.; and Tambe, M. 2019. End to end learning and optimization on graphs. *Advances in Neural Information Processing Systems*, 32: 4672–4683.

Xie, Y.; Dai, H.; Chen, M.; Dai, B.; Zhao, T.; Zha, H.; Wei, W.; and Pfister, T. 2020. Differentiable top-k with optimal transport. *Advances in Neural Information Processing Systems*, 33: 20520–20531.

Yahoo! 2007. Webscope dataset. Ydata-ysm-advertiser-bids-v1.0.