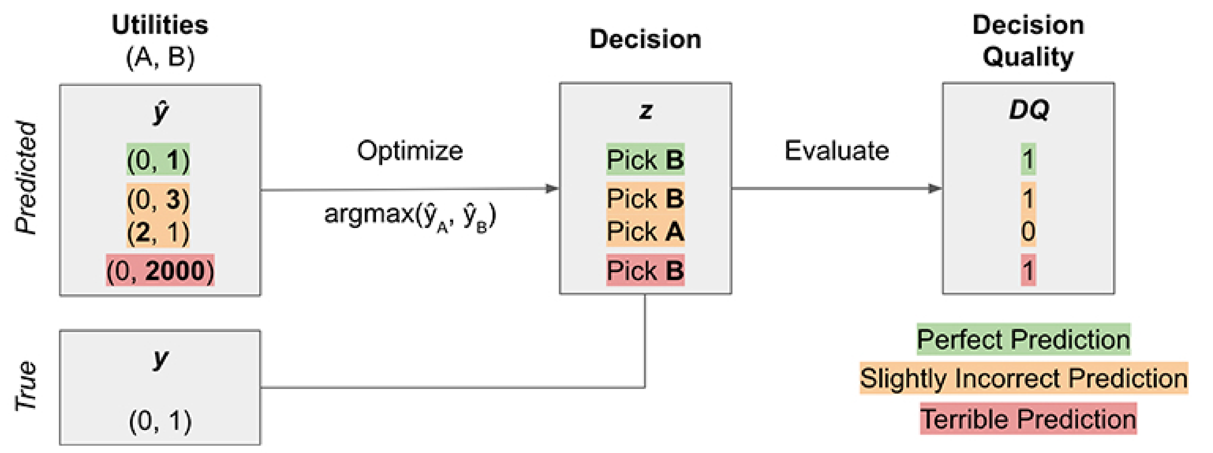
Decision-Focused Learning without Differentiable Optimization: Learning Locally Optimized Decision Losses

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

TL;DR: We learn task-specific and convex loss functions (LODLs) by approximating the Decision Loss in Predict-Then-Optimize problems using samples. These LODLs lead to better Decision Quality in three domains from the literature!

Motivation

Consider the following Predict-Then-Optimize problem: (1) **Predict** the utility of A and B receiving a resource respectively, and (2) use these predictions to **optimize** for an allocation that maximizes social welfare. Finally, evaluate the "**Decision Quality**" of this allocation – how well the allocation would have done in the real world (with the true utilities).

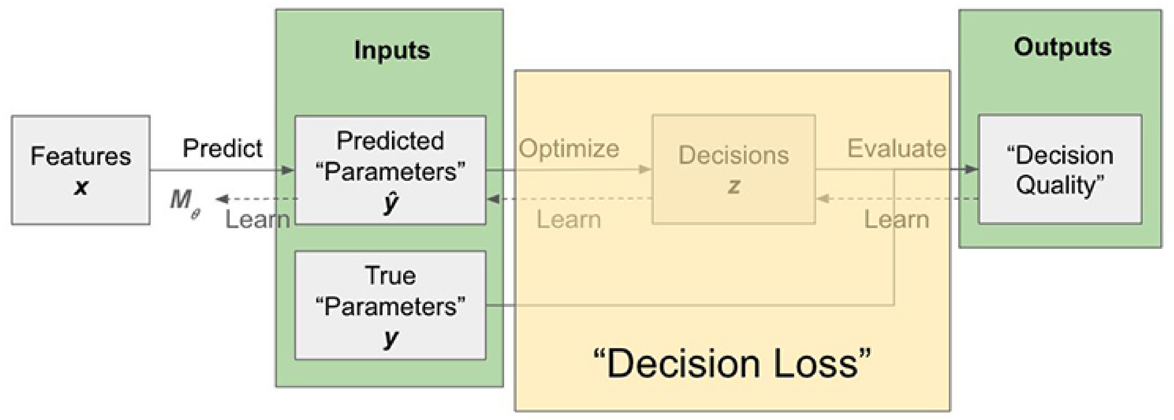


Problem With "Standard" Losses

There's a **mismatch** between what predictive models are optimized for (e.g. Mean Squared Error) and what it's evaluated using (Decision Quality). For example, in the figure above, **both "slightly incorrect predictions" have the same MSE, but different DQs.**

Main Idea

The Predict-Then-Optimize pipeline can be interpreted as being a loss in itself. While the actual form of the **Decision Loss (DL)** is complex, we use **supervised learning** to approximate this mapping using samples.

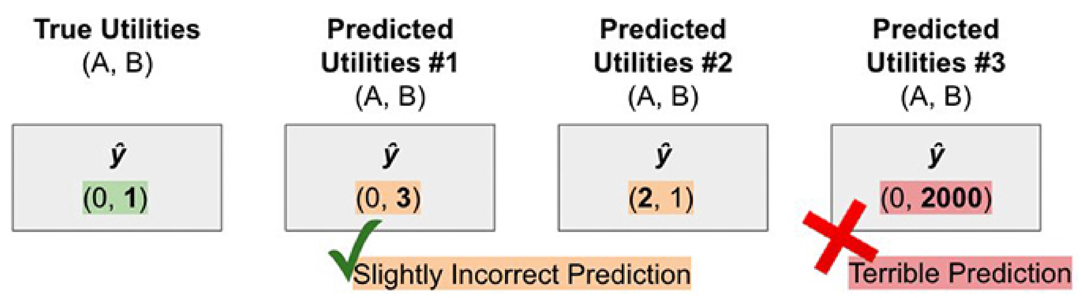


Method

There are 2 big challenges to learning DL. First, to learn the decision loss, we need the "Predicted Parameters" as input. However, to train the predictive model that generates these parameters, we need a loss function. This leads to a **chicken-and-egg problem**. Second, given a dataset, how should we **parameterize the loss**?

Challenge 1: Sampling "Predicted Parameters"

To resolve the chicken-and-egg problem, we posit a **"localness" assumption** that says that the predictive model will always get you **close** to the true predictions. As a result, we only need to sample from the distribution of **"slightly incorrect predictions"**. Concretely, we assume that the predicted parameters are the true parameters plus some Gaussian noise.



$$\hat{\boldsymbol{y}}_i = \boldsymbol{y}_i + \mathcal{N}(0, \boldsymbol{I})$$

Challenge 2: Learning Convex Loss Functions

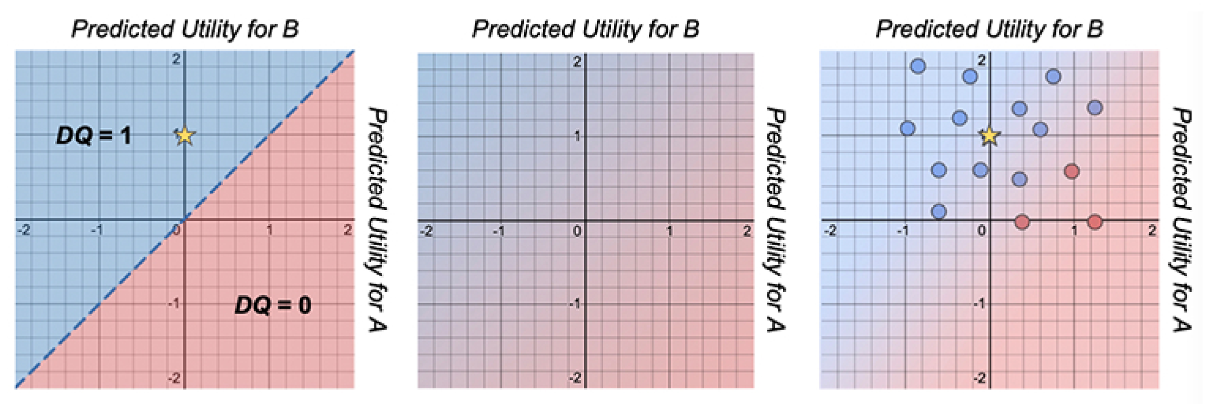
We posit 2 sets of low-dimensional loss functions for each set of predictions. These are **easy to learn** and **convex-by-construction**.

$$\text{WeightedMSE}(\hat{\boldsymbol{y}}) = \sum_{l=1}^{\dim(\boldsymbol{y})} w_l \cdot (\hat{y}_l - y_l)^2 \qquad w_l = \begin{cases} w_+, & \text{if } \hat{y}_i - y_i \geq 0 \\ w_-, & \text{otherwise} \end{cases}$$

$$\text{Quadratic}(\hat{\boldsymbol{y}}) = (\hat{\boldsymbol{y}} - \boldsymbol{y})^T H (\hat{\boldsymbol{y}} - \boldsymbol{y}) \qquad H = L^T L \qquad L_{ij} = \begin{cases} L_{ij}^{++}, & \text{if } \hat{y}_i - y_i \geq 0 \text{ and } \hat{y}_j - y_j \geq 0 \\ L_{ij}^{+-}, & \text{if } \hat{y}_i - y_i \geq 0 \text{ and } \hat{y}_j - y_j < 0 \\ L_{ij}^{-+}, & \text{if } \hat{y}_i - y_i < 0 \text{ and } \hat{y}_j - y_j \geq 0 \\ L_{ij}^{--}, & \text{otherwise} \end{cases}$$

Putting it Together

For the example we described earlier, the **true DL is on the left**. **Our method is on the right**. **In the middle is the "Decision-Focused Learning" (DFL) baseline** from the literature. There, they propose a handcrafted relaxation of the optimization problem that leads to smooth (but possibly non-convex) DLs.



True Optimization | "Surrogate" Optimization | Learned Loss (Without Handcrafting)

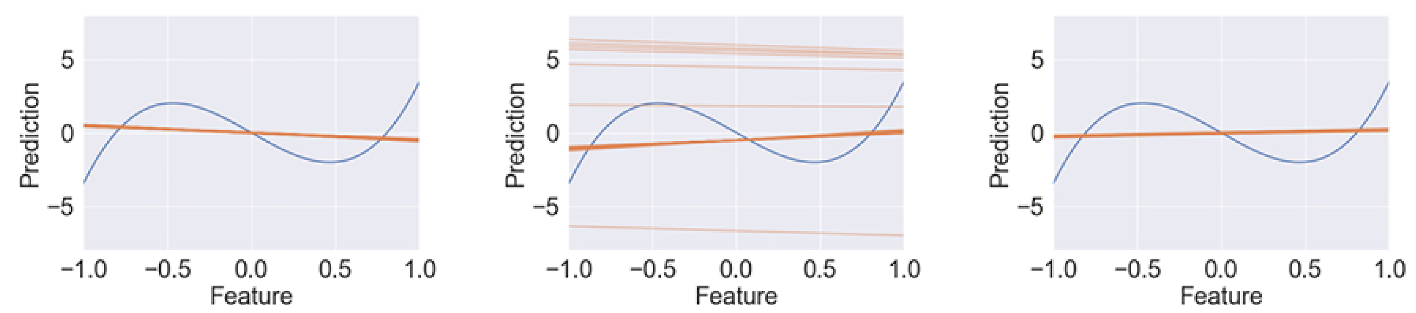
Experiments

We evaluate the performance of our loss functions on three domains from the literature. As baselines, we use a "standard" loss (MSE) as well as the DFL method proposed by the paper that created the domain. **We find that DirectedQuadratic consistently outperforms 2-stage!**

Loss Function	Normalized DQ On Test Data		
	Linear Model	Web Advertising	Portfolio Optimization
Random	0	0	0
Optimal	1	1	1
2-Stage (MSE)	-0.953 ± 0.000	0.476 ± 0.147	0.320 ± 0.015
DFL	0.828 ± 0.383	0.854 ± 0.100	**0.348 ± 0.015**
NN	**0.962 ± 0.000**	0.814 ± 0.137	-0.105 ± 0.084
WeightedMSE	-0.934 ± 0.060	0.576 ± 0.151	0.308 ± 0.018
DirectedWeightedMSE	**0.962 ± 0.000**	0.533 ± 0.137	0.322 ± 0.015
Quadratic	-0.752 ± 0.377	**0.931 ± 0.040**	0.272 ± 0.020
DirectedQuadratic	**0.962 ± 0.000**	0.910 ± 0.043	0.325 ± 0.014

Visualizing Learned Models

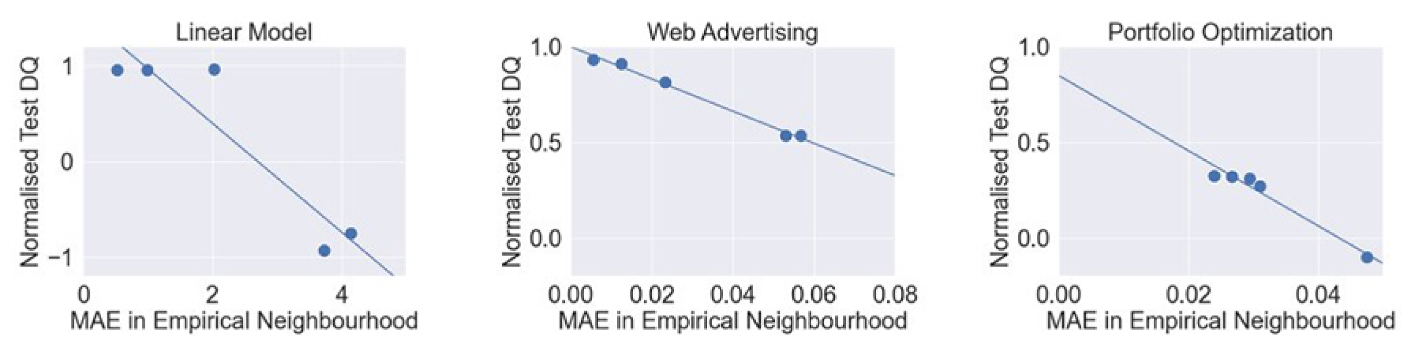
The Linear Model domain is an extension of our earlier example. The blue line corresponds to the true mapping between the features and utilities; we try to fit a linear model to approximate this relationship. **MSE focuses on minimizing the error on all the points** and leads to a negative slope. However, because **we only care about the maximum utility element**, the decision-aware models lead to a positive slope!



(a) 2-Stage | (b) DFL | (c) DirectedQuadratic

Better Losses Lead To Better Models

The better our losses model the true Decision Loss (x-axis), the better the predictive models learned with that loss (y-axis)!



1 | 2 MACHINE LEARNING DEPARTMENT | 3

On the job market!