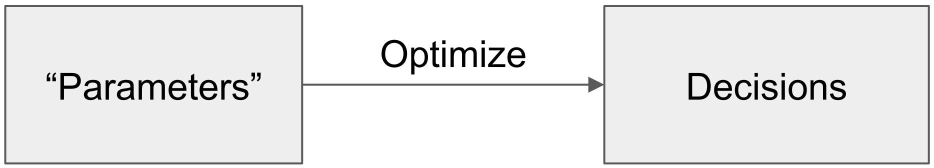
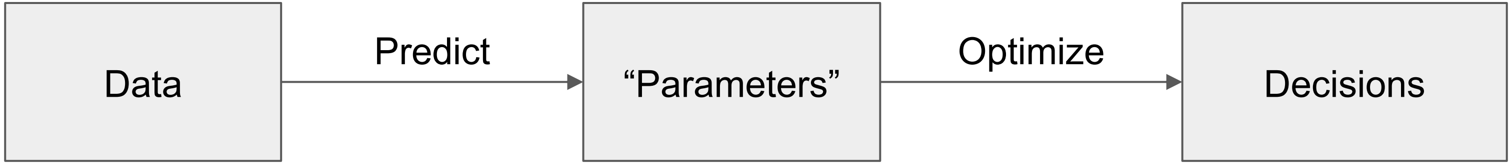
Decision-Focused Learning without Decision-Making: Learning Locally Optimized Decision Losses

Sanket Shah

Motivation: ML-based Decision Making

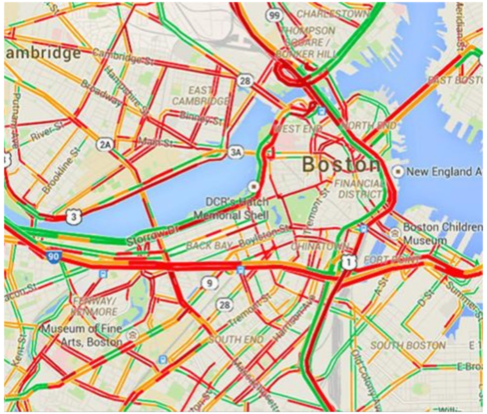


Motivation: ML-based Decision Making



Motivation: ML-based Decision Making

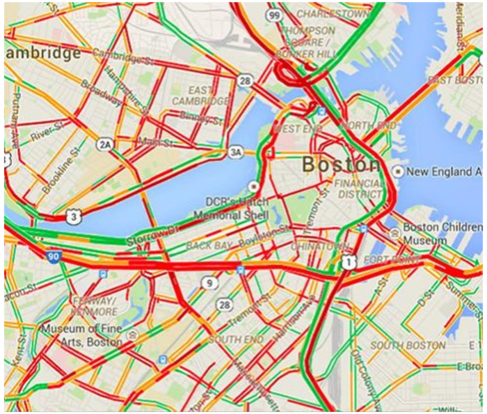
Data $\xrightarrow{\text{Predict}}$ "Parameters" $\xrightarrow{\text{Optimize}}$ Decisions



Route Planning

Motivation: ML-based Decision Making

Data $\xrightarrow{\text{Predict}}$ “Parameters” $\xrightarrow{\text{Optimize}}$ Decisions

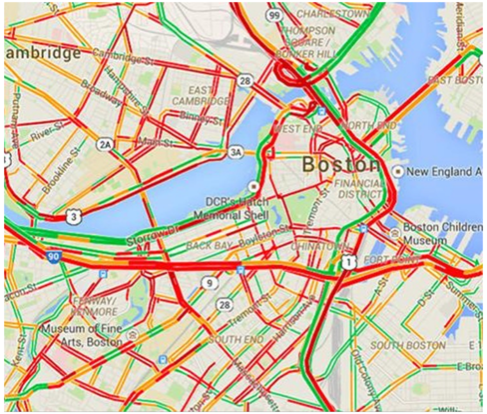


Route Planning

Wildlife Conservation

Motivation: ML-based Decision Making

Data —Predict→ "Parameters" —Optimize→ Decisions



Route Planning

Wildlife Conservation

Public Health

Example: ARMMAN

- **mMitra:** Maternal health information via voice and text messages (*2.6 million women reached!*)
  - **Limited Resource:** Phone call by health worker

Example: ARMMAN

- **mMitra:** Maternal health information via voice and text messages (*2.6 million women reached!*)
  - **Limited Resource:** Phone call by health worker

Age: 23
Location: Rajasthan
Education: Grade 10

Age: 28
Location: Rajasthan
Education: Undergrad

Example: ARMMAN

- **mMitra:** Maternal health information via voice and text messages (*2.6 million women reached!*)
  - **Limited Resource:** Phone call by health worker

Age: 23
Location: Rajasthan
Education: Grade 10

Predict →

25%
Prob of Dropping Off

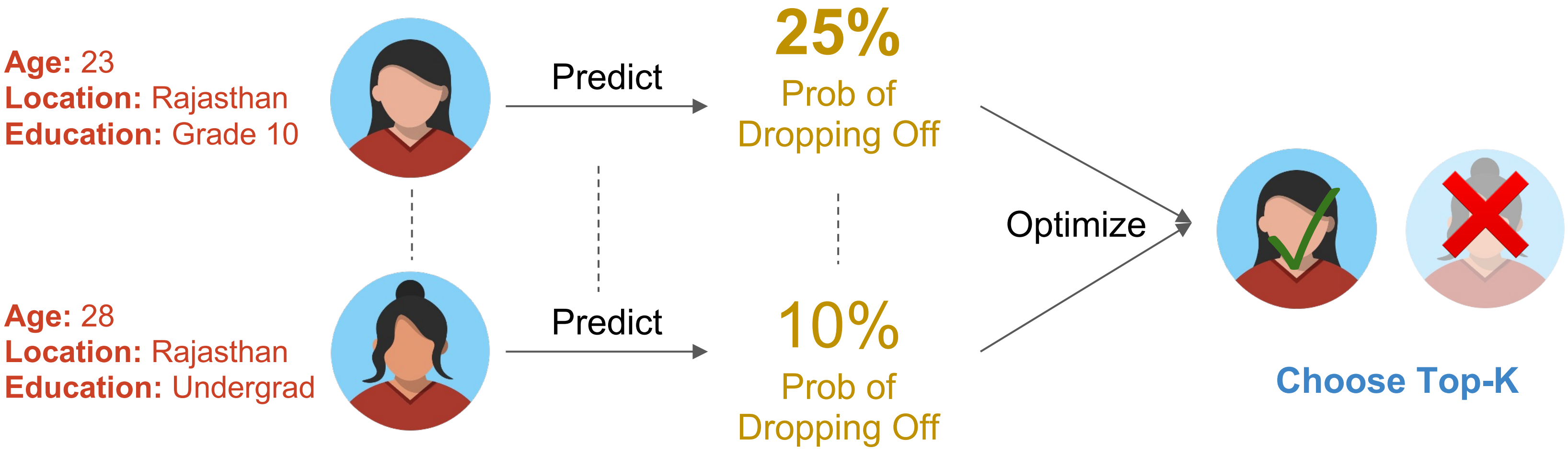
Age: 28
Location: Rajasthan
Education: Undergrad

Predict →

10%
Prob of Dropping Off

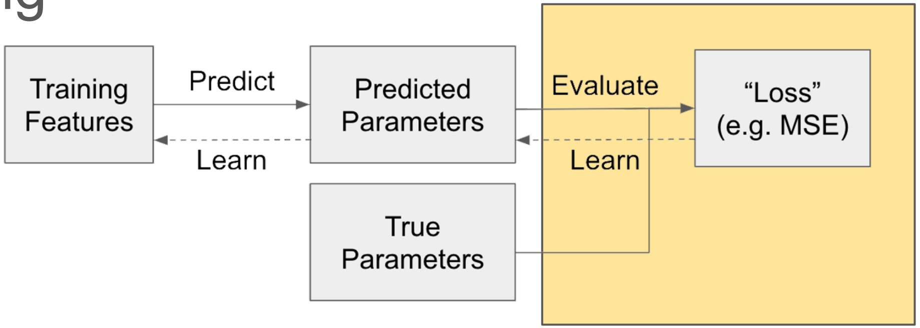
Example: ARMMAN

- **mMitra:** Maternal health information via voice and text messages (*2.6 million women reached!*)
    - **Limited Resource:** Phone call by health worker



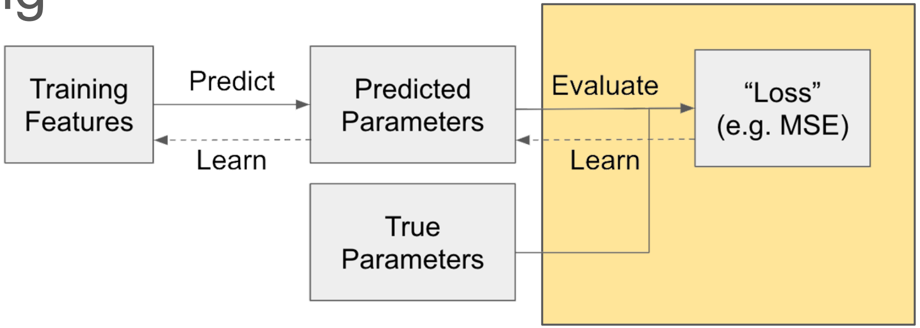
Standard Solution: “2-Stage” Learning

- Training

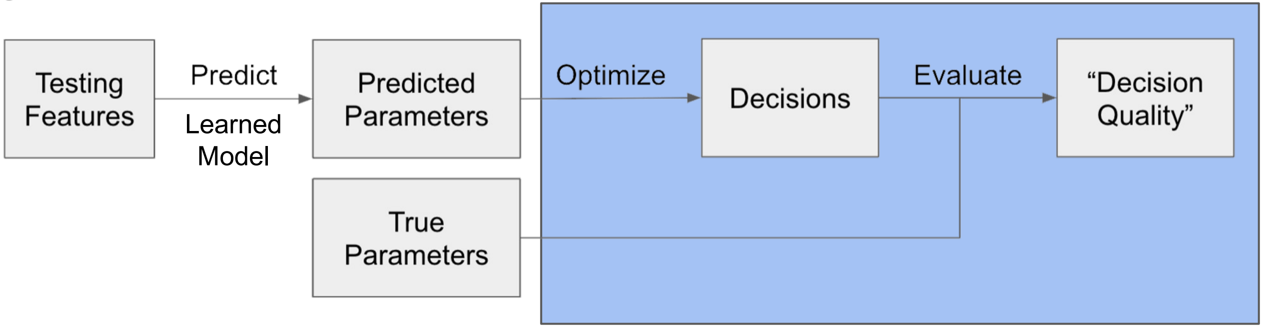


Standard Solution: “2-Stage” Learning

- Training

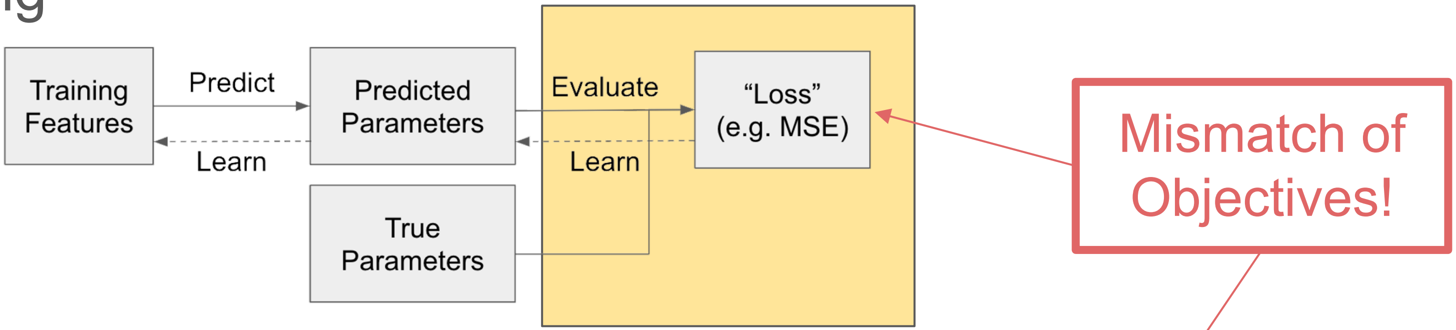


- Testing

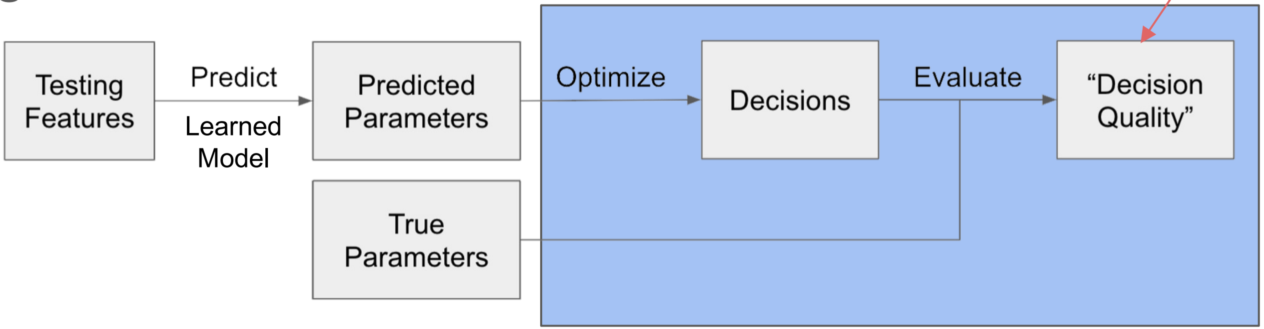


Standard Solution: "2-Stage" Learning

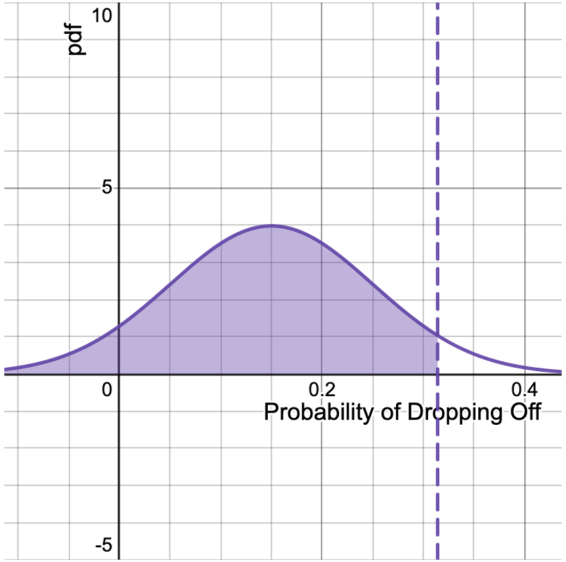
- Training



- Testing



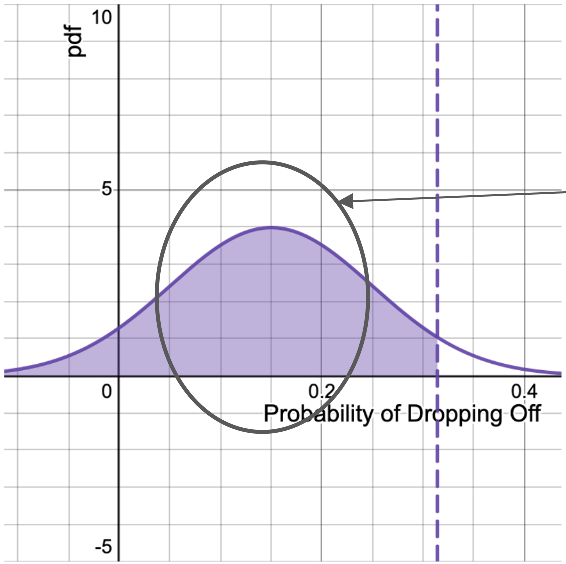
Mismatch of Objectives



True Distribution

Objective: Choose top 5% of beneficiaries

Mismatch of Objectives

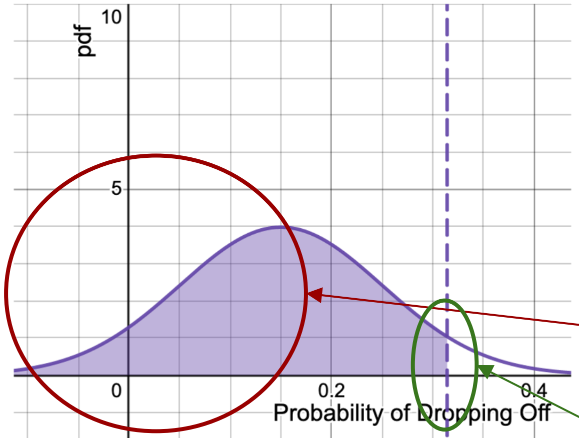


True Distribution

Objective: Choose top 5% of beneficiaries

MSE: Focus on getting *most* of the predictions approximately right.

Mismatch of Objectives



True Distribution

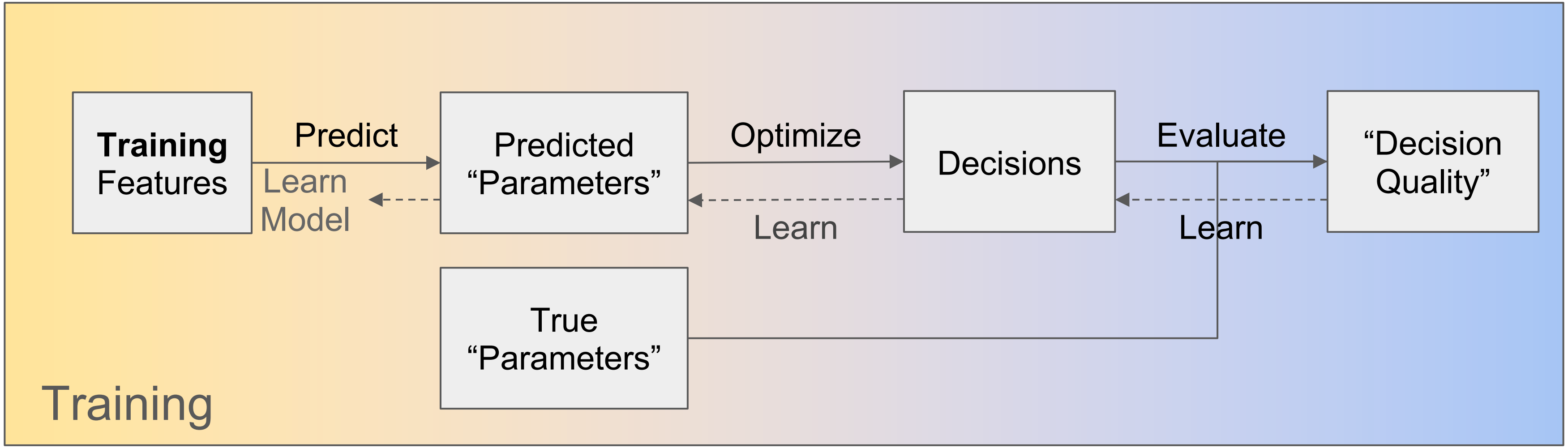
Objective: Choose top 5% of beneficiaries

MSE: Focus on getting *most* of the predictions approximately right.

But: It's okay to make errors on these predictions…

As long as you get these predictions correct!

SoTA: Decision Focused Learning (DFL)

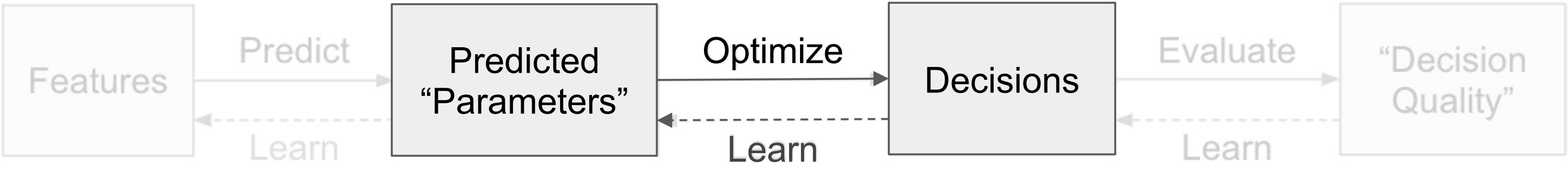


We can learn better models by taking into account task structure while training!

[Elmachtoub and Grigas 2022, Donti et al. 2017, Wilder et al. 2019]

Challenge

- Differentiating through the optimization problem is difficult:

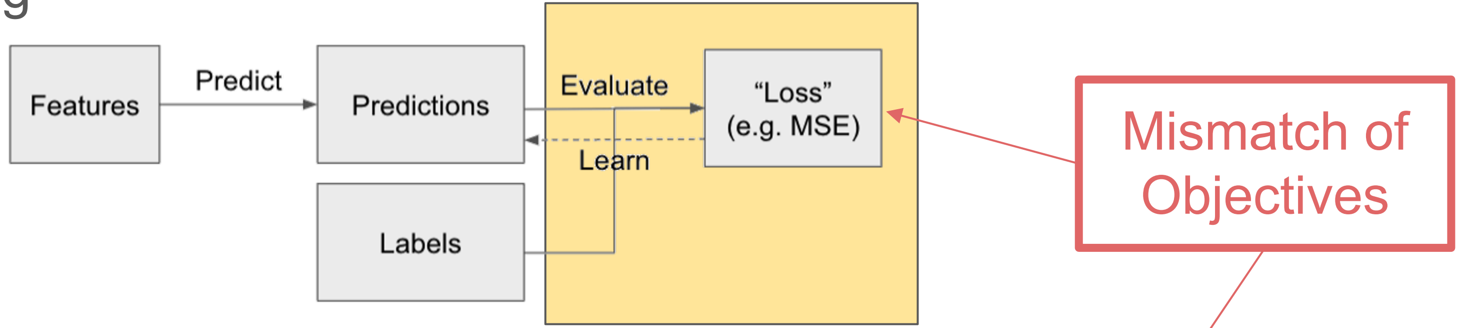


  - E.g., argmax operation is non-smooth in discrete optimization

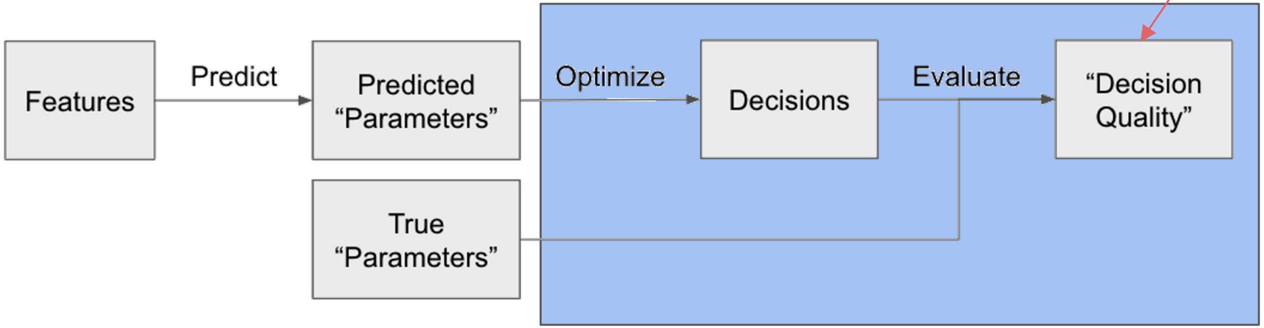
- **Past Work:** Create "surrogate" problems that you *can* differentiate through. **BUT:**
  A. Surrogates are *handcrafted* and *task-specific*
  B. Surrogates are often *not convex*

Contribution (1)

- Training

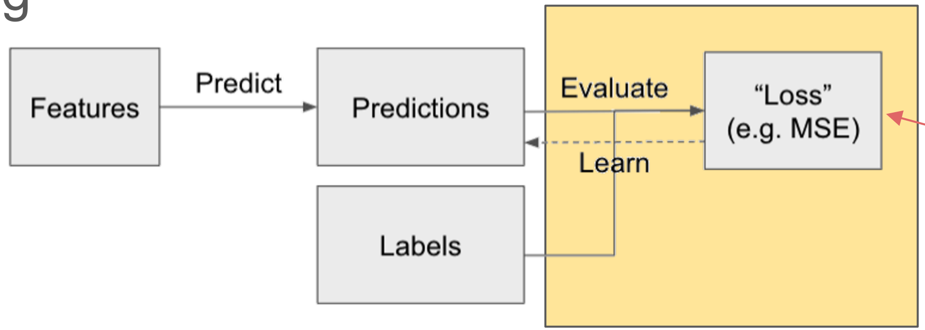


- Testing

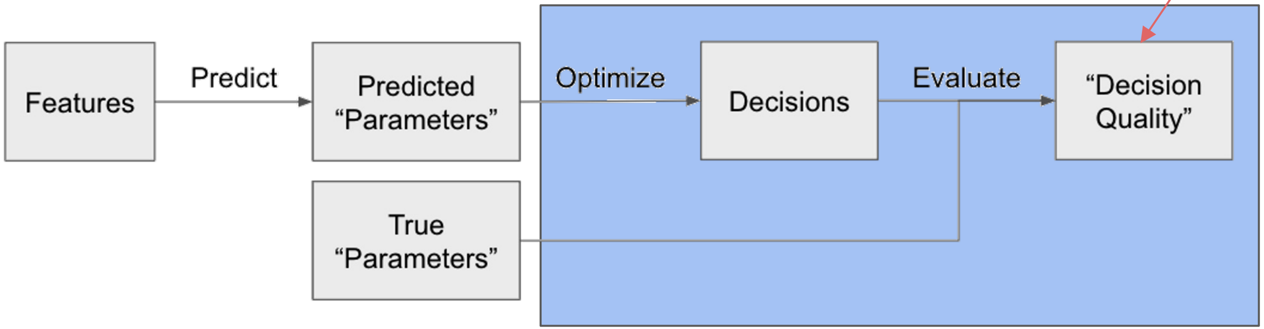


Contribution (1)

- Training



- Testing



Align "Loss" and "Decision Quality" using **Task-Specific Loss Functions**

Contribution (2)

- **Idea:** (A) *Automatically learn* task-specific “loss” functions that are (B) *convex-by-construction*
  - Does away with argmax/surrogates altogether!

- **Results:** We outperform 2-stage on three resource allocation domains from the literature
  - We even do better than DFL in the two domains where DFL requires surrogates!

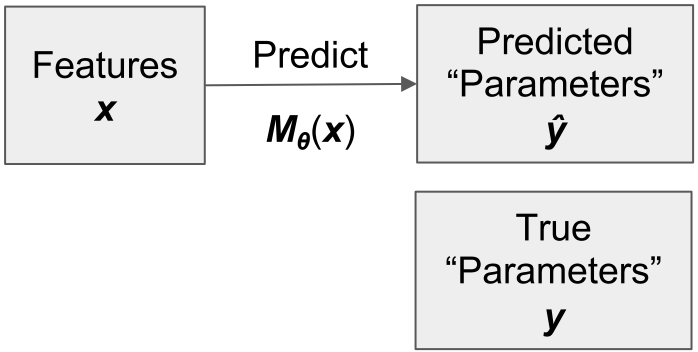
Outline

- Introduction
- **Predict-Then-Optimize Details**
- Our Approach
- Experiments
- Conclusions and Future Work

Notation

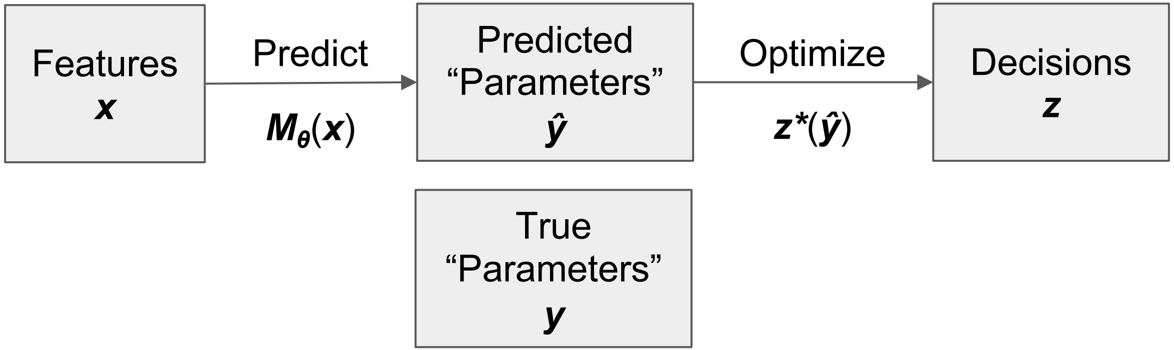
Features
x

Notation



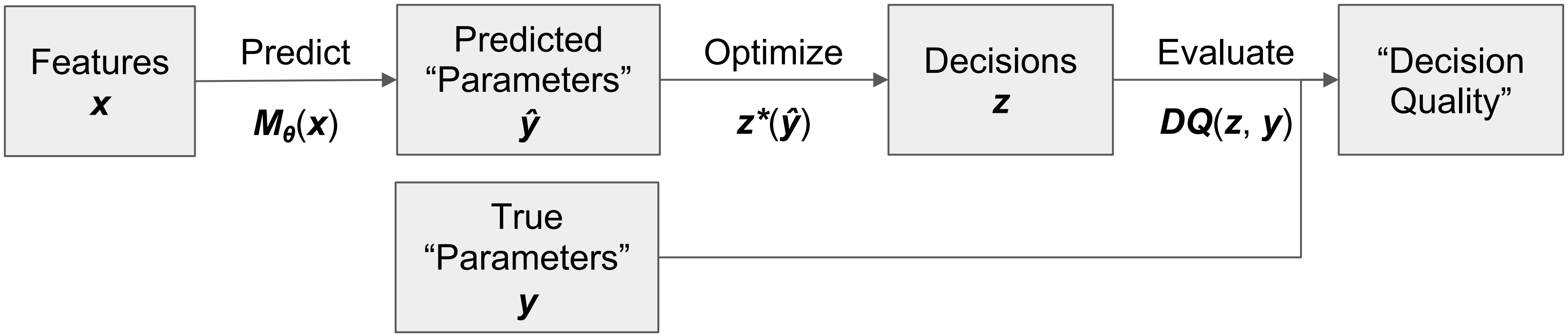
Features
$\boldsymbol{x}$
Predict
$\boldsymbol{M_\theta}(\boldsymbol{x})$
Predicted
“Parameters”
$\boldsymbol{\hat{y}}$
True
“Parameters”
$\boldsymbol{y}$

Notation



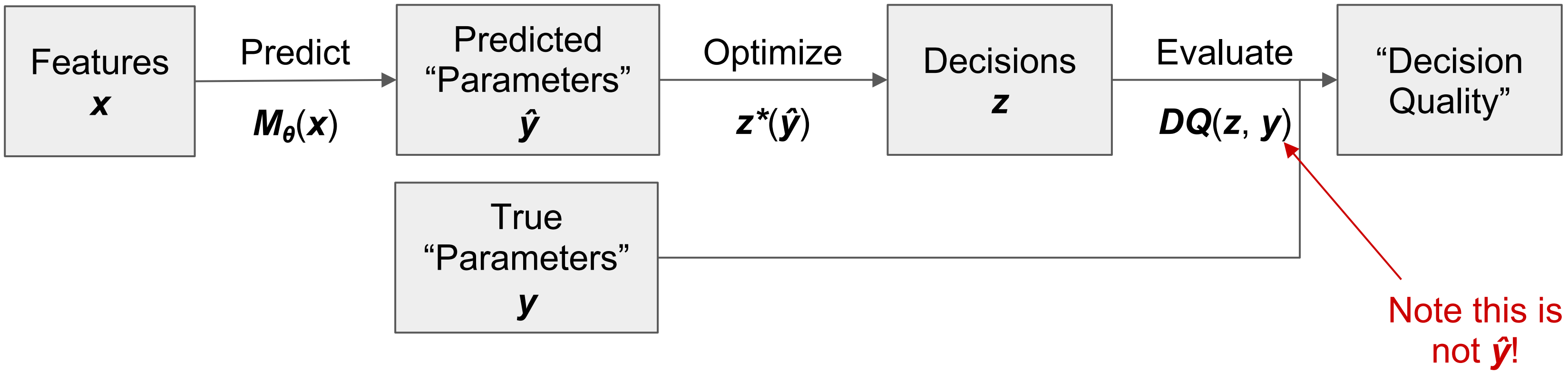
Features
$\boldsymbol{x}$
Predict
$\boldsymbol{M_\theta}(\boldsymbol{x})$
Predicted
"Parameters"
$\boldsymbol{\hat{y}}$
Optimize
$\boldsymbol{z^*}(\boldsymbol{\hat{y}})$
Decisions
$\boldsymbol{z}$
True
"Parameters"
$\boldsymbol{y}$

Notation

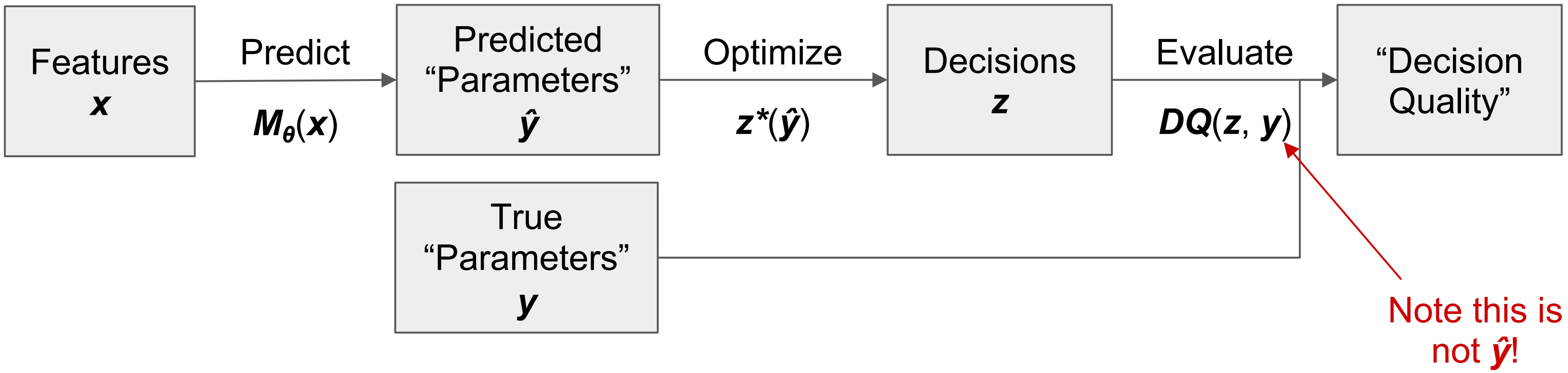


Features
$\boldsymbol{x}$
Predict
$\boldsymbol{M_\theta}(\boldsymbol{x})$
Predicted "Parameters"
$\boldsymbol{\hat{y}}$
Optimize
$\boldsymbol{z^*}(\boldsymbol{\hat{y}})$
Decisions
$\boldsymbol{z}$
Evaluate
$\boldsymbol{DQ}(\boldsymbol{z}, \boldsymbol{y})$
"Decision Quality"
True "Parameters"
$\boldsymbol{y}$

Notation


Features
x
Predict
$M_\theta(x)$
Predicted
“Parameters”
$\hat{y}$
Optimize
$z^*(\hat{y})$
Decisions
z
Evaluate
$DQ(z, y)$
“Decision
Quality”
True
“Parameters”
y
Note this is
not $\hat{y}$!

Notation



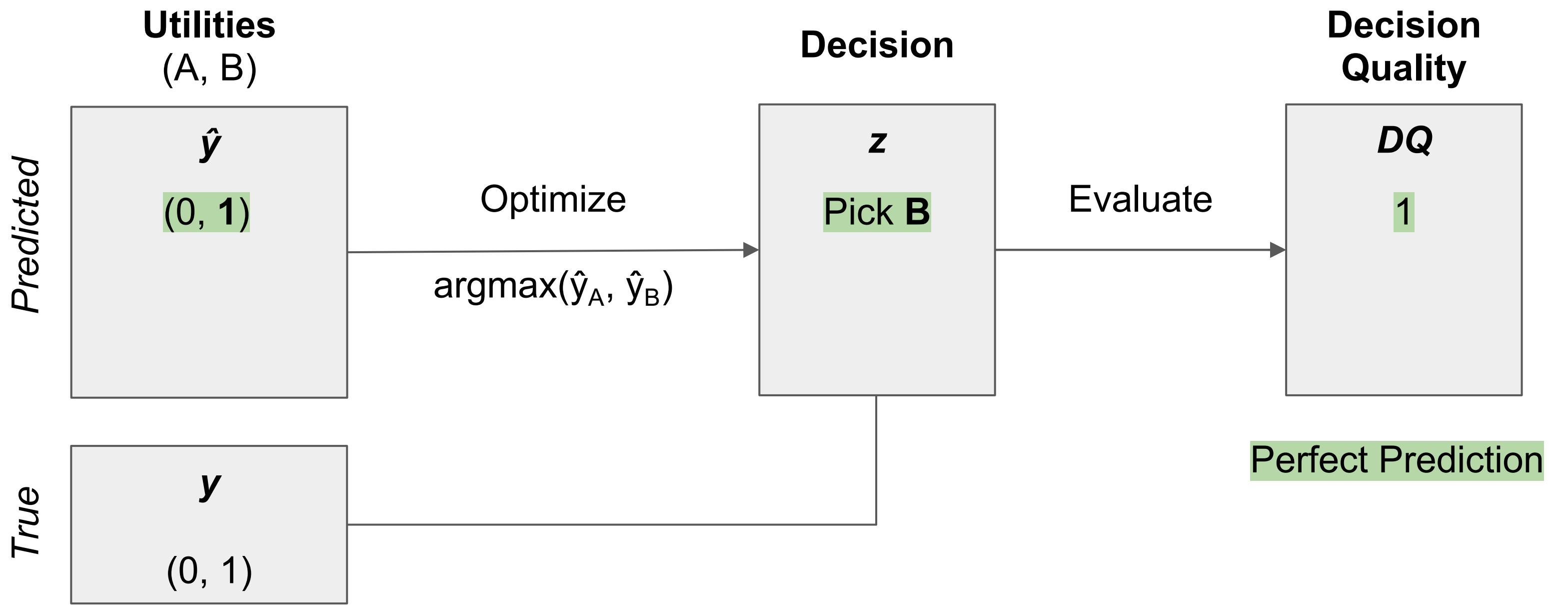
- **Decision Quality:** How good are the ***decisions*** made on *predicted* parameters when tested on the ***true*** parameters?

Minimal Example

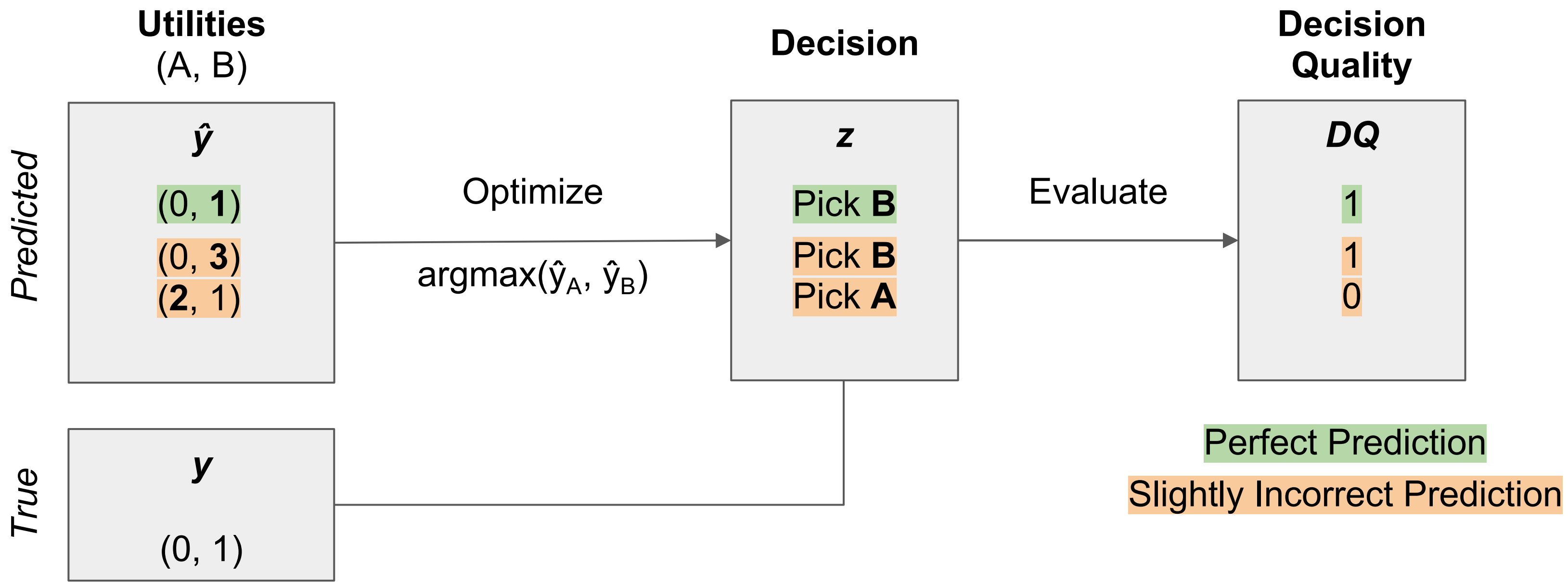
- **Resource Allocation:** 2 Beneficiaries (A and B), 1 Resource
  - **Predict:** *Utilities* for beneficiaries
  - **Optimize:** Give resource to beneficiary with higher utility

- **Decision Quality:** True utility of the beneficiary who you give the resource to

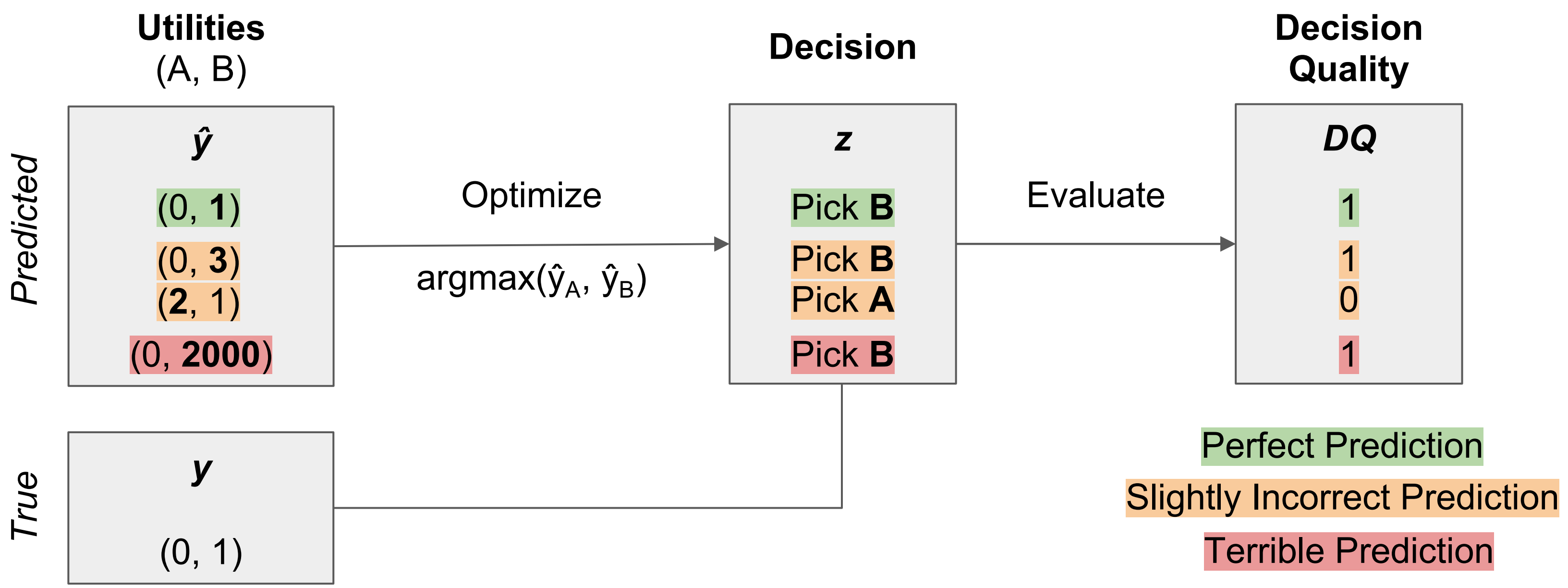
Prediction Accuracy vs. Decision Quality



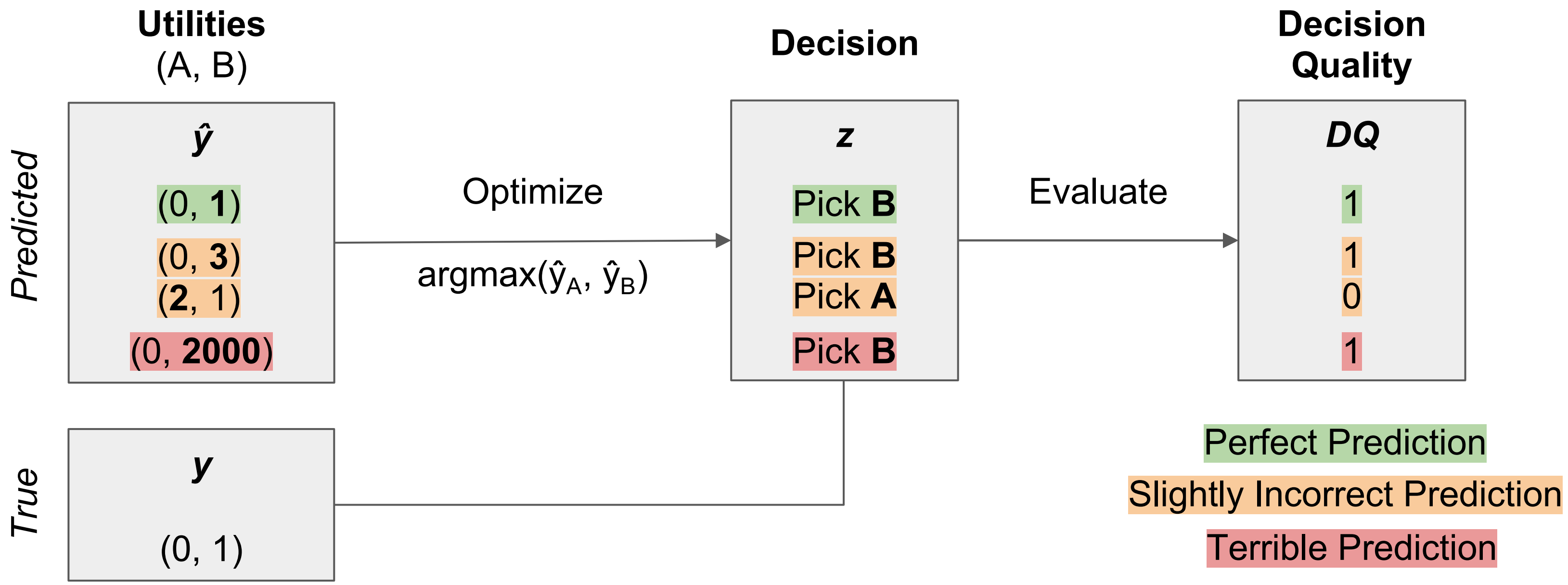
Prediction Accuracy vs. Decision Quality



Prediction Accuracy vs. Decision Quality

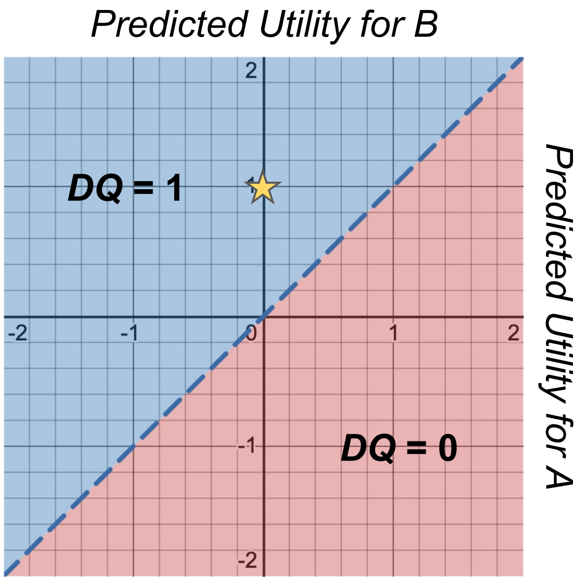


Prediction Accuracy vs. Decision Quality

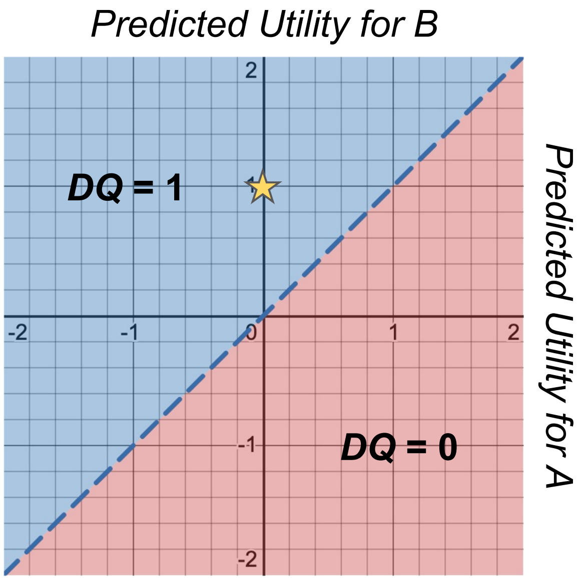


Predictive Accuracy $\not\propto$ **sion Quality**

Non-Smoothness



Non-Smoothness

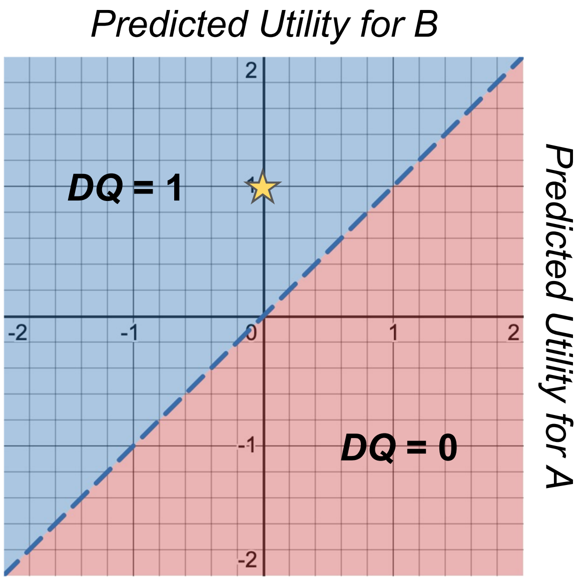


Decision quality is piecewise constant

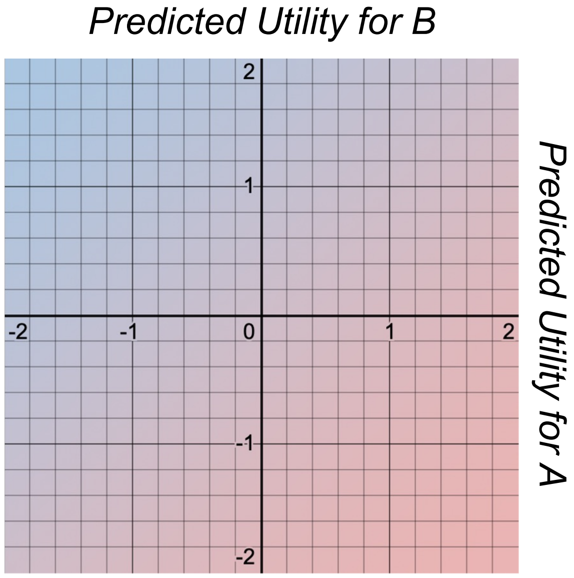
↓

Gradients for DFL are uninformative

Non-Smoothness



True Optimization

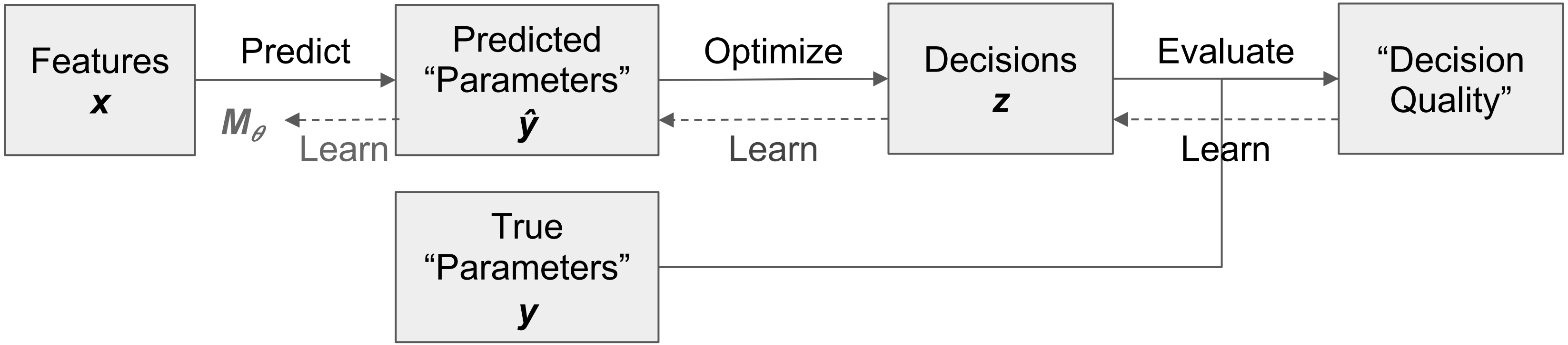


"Surrogate" Optimization

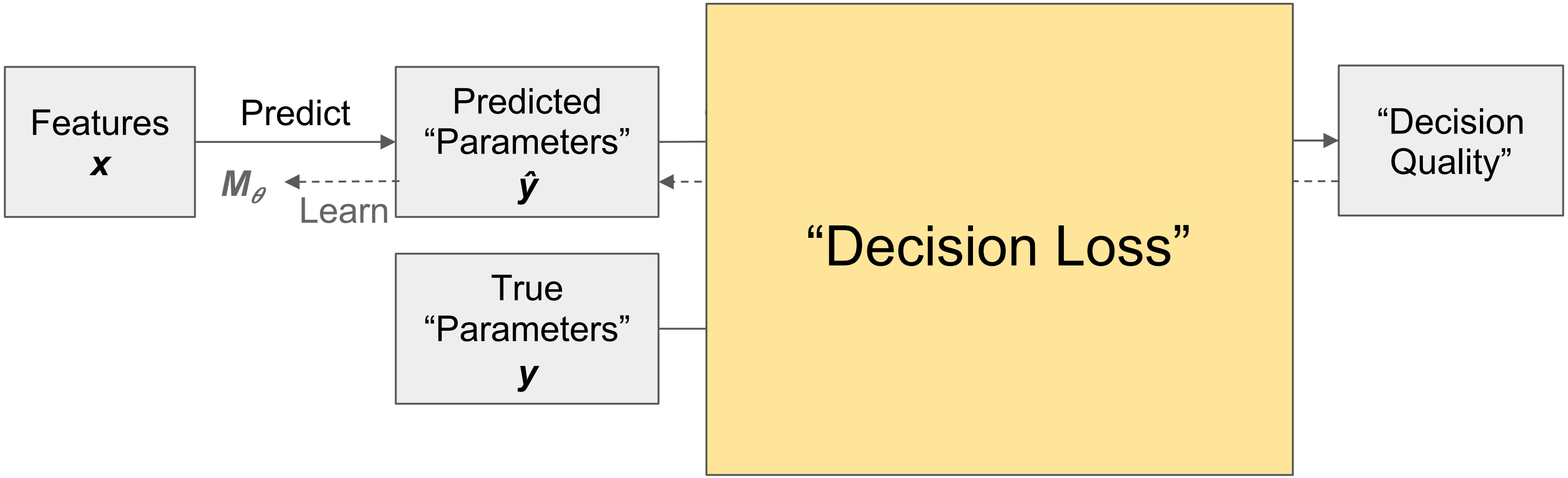
Outline

- Introduction
- Predict-Then-Optimize Details
- **Our Approach**
- Experiments
- Conclusions and Future Work

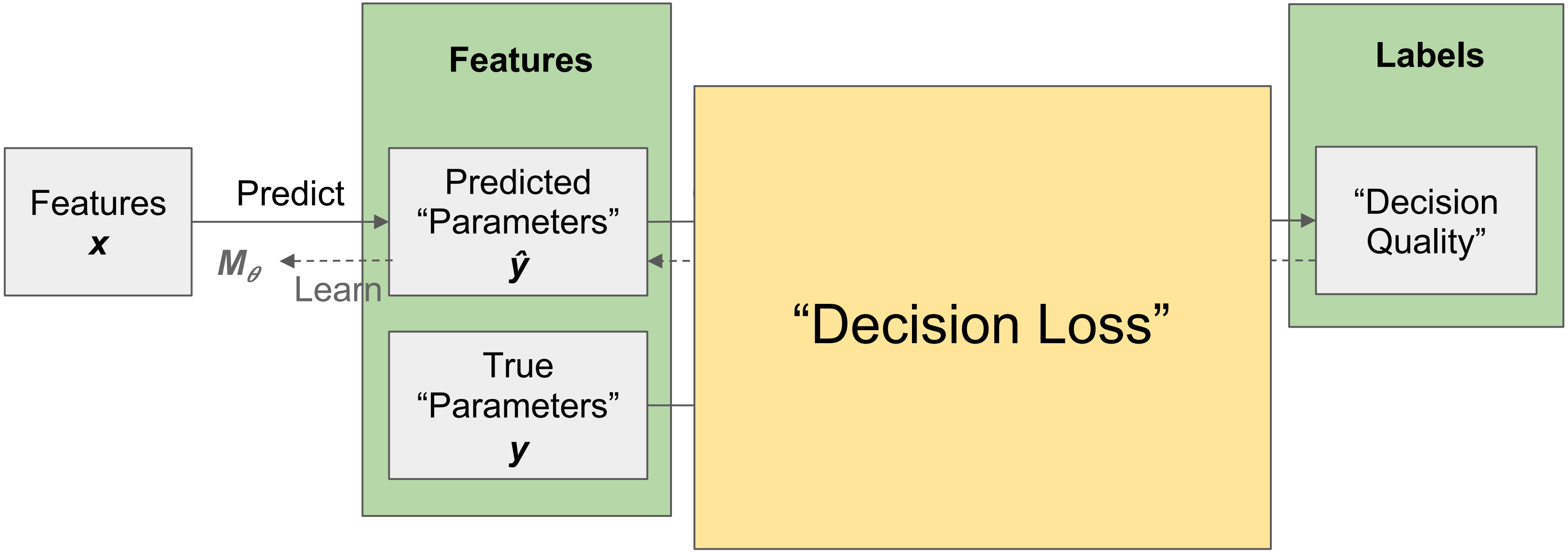
“Decision Loss”



"Decision Loss"

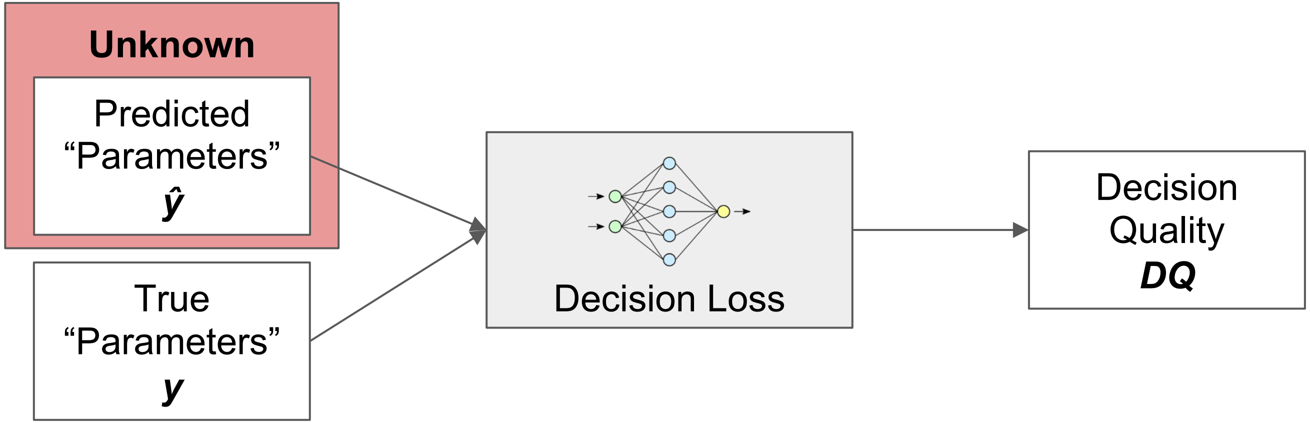


“Decision Loss”



We formulate learning the decision loss as a supervised learning problem

Learning Decision Loss



- **Step 1:** Generate samples of "realistic" ($\hat{y}$, y) inputs and calculate ***DQ*** to create training data
- **Step 2:** Fit a *convex-by-construction* model to these input-output pairs

Step 1: Generate "Predicted Parameters"

Step 1: Generate “Predicted Parameters”

But how? Don’t we need a predictive model for that?

Localness Assumption

- **The predictive model will get you *close* to the true params**
  - Decision Loss' job is to help differentiate between predictions that are close to the true label
  - "Realistic predictions" → "Approximately correct predictions"

Localness Assumption

- **The predictive model will get you *close* to the true params**
  - Decision Loss' job is to help differentiate between predictions that are close to the true label
  - "Realistic predictions" → "Approximately correct predictions"

True Utilities
(A, B)

$\hat{y}$
(0, **1**)

Localness Assumption

- **The predictive model will get you *close* to the true params**
  - Decision Loss' job is to help differentiate between predictions that are close to the true label
  - "Realistic predictions" → "Approximately correct predictions"

True Utilities (A, B)	**Predicted Utilities #1** (A, B)	**Predicted Utilities #2** (A, B)
$\hat{y}$ (0, **1**)	$\hat{y}$ (0, **3**)	$\hat{y}$ (**2**, 1)

✓ Slightly Incorrect Prediction

Localness Assumption

- **The predictive model will get you *close* to the true params**
  - Decision Loss' job is to help differentiate between predictions that are close to the true label
  - "Realistic predictions" → "Approximately correct predictions"

True Utilities (A, B)	**Predicted Utilities #1** (A, B)	**Predicted Utilities #2** (A, B)	**Predicted Utilities #3** (A, B)
$\hat{y}$	$\hat{y}$	$\hat{y}$	$\hat{y}$
(0, **1**)	(0, **3**)	(**2**, 1)	(0, **2000**)

Slightly Incorrect Prediction

Terrible Prediction

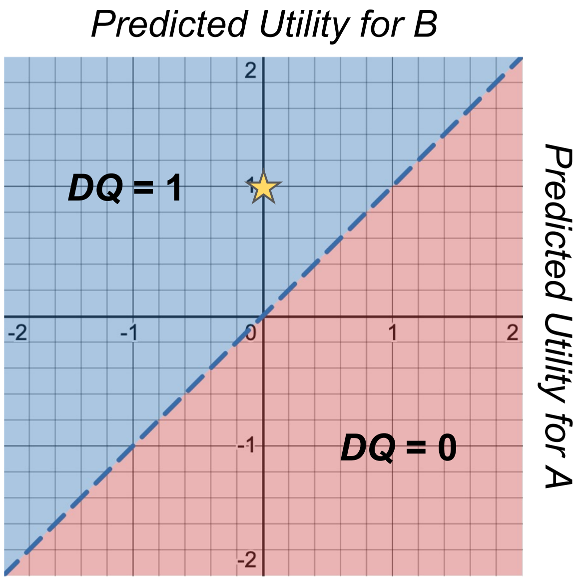
Sampling Strategies (Step 1)

- Sample “realistic”/nearby points *by adding Gaussian Noise to the true parameters*:
  - All-Perturbed: Add noise to *all n* dimensions simultaneously

$$\boldsymbol{y}_n^i = \boldsymbol{y}_n + \boldsymbol{\epsilon}^k = \boldsymbol{y}_n + \alpha \cdot \mathcal{N}(0, I)$$

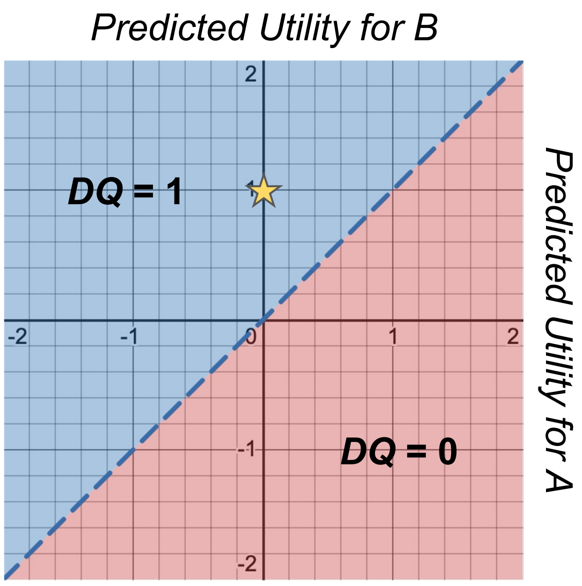
  - 1-Perturbed and 2-Perturbed: Perturb 1 or 2 dimensions at a time.
    - Similar to calculating the numerical gradient and hessian

In the context of past approaches

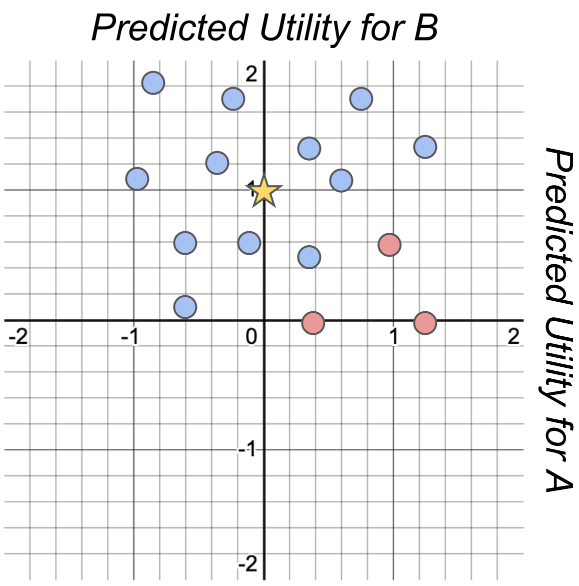


True Optimization

In the context of past approaches

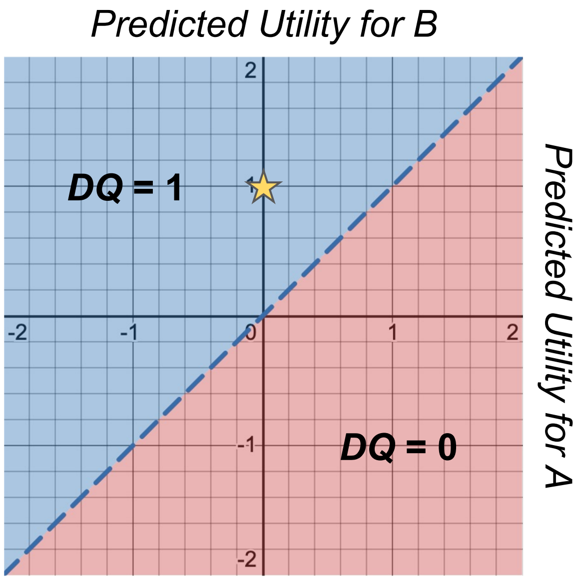


True Optimization

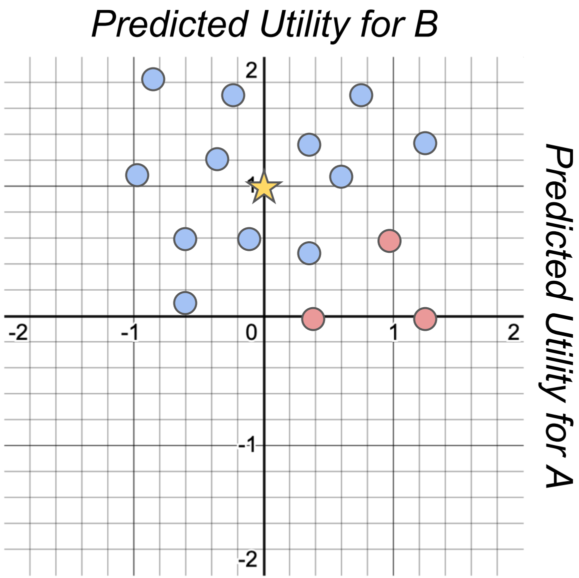


Sampled Points

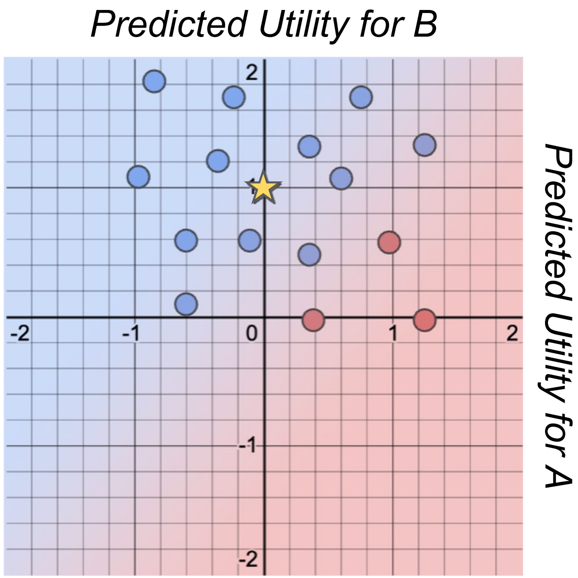
In the context of past approaches



True Optimization

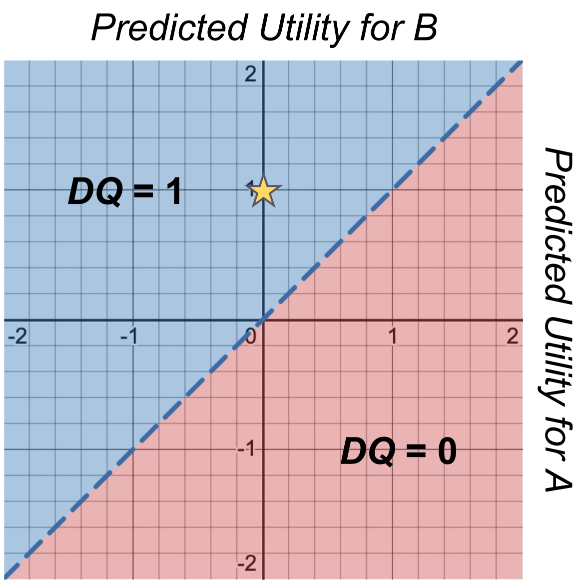


Sampled Points

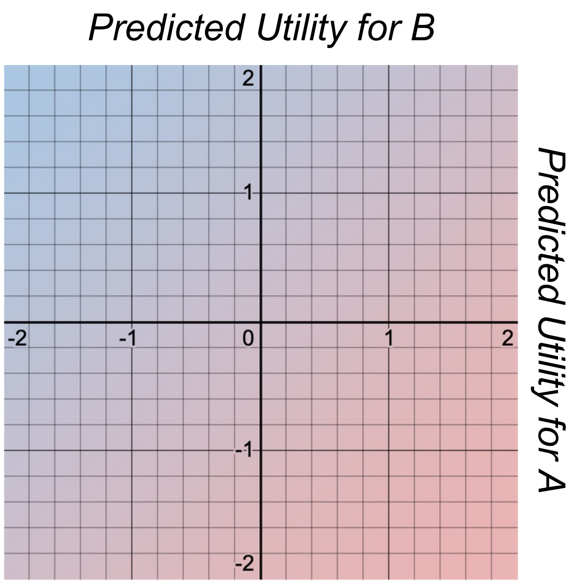


Learned Loss
(Without Handcrafting)

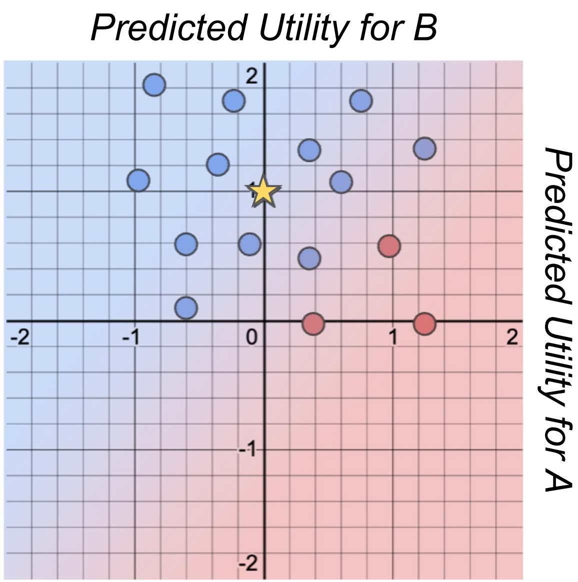
In the context of past approaches



True Optimization



“Surrogate” Optimization



Learned Loss
(Without Handcrafting)

Step 2: Learn a Task-Specific Loss

Step 2: Learn a Task-Specific Loss

How do we make it “convex-by-construction”?

Loss Function Families (Step 2)

- (Approach 1) **Weighted-MSE:**
  - Hypothesis: Decision Quality is not equally sensitive to all parameters

True Utilities
(A, B)

$\hat{\boldsymbol{y}}$
(0, 1)

Loss Function Families (Step 2)

- (Approach 1) **Weighted-MSE:**
  - <u>Hypothesis:</u> Decision Quality is not equally sensitive to all parameters

True Utilities (A, B)		**True Utilities** (A, B, C, D)
$\hat{y}$ (0, 1)	→	$\hat{y}$ (0, 1, 1000, **1001**)

Loss Function Families (Step 2)

- (Approach 1) **Weighted-MSE:**
  - Hypothesis: Decision Quality is not equally sensitive to all parameters

True Utilities
(A, B)

$\hat{y}$

(0, 1)

→

True Utilities
(A, B, C, D)

$\hat{y}$

(0, 1, 1000, **1001**)

Loss Function Families (Step 2)

- (Approach 1) **Weighted-MSE:**
  - Hypothesis: Decision Quality is not equally sensitive to all parameters

True Utilities
(A, B)

$\hat{y}$

(0, 1)

→

True Utilities
(A, B, C, D)

$\hat{y}$

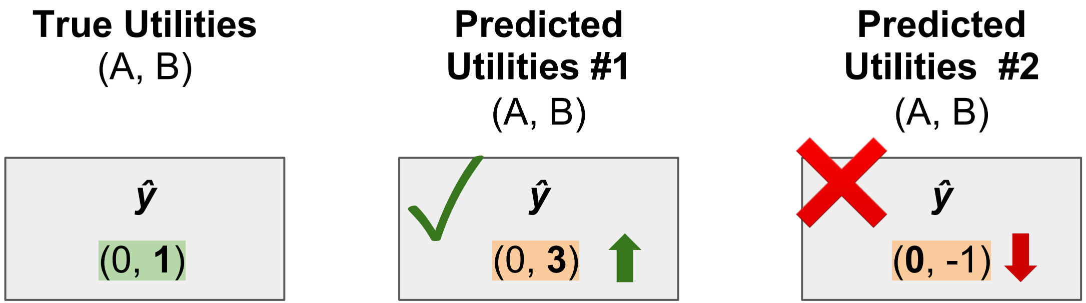
(0, 1, 1000, **1001**)

  - Idea: Learn a "weight" for each parameter, based on how much it affects the Decision Quality

$$\sum_{l=1}^{dim(\boldsymbol{y})} w_l \cdot (\hat{\boldsymbol{y}}_l - \boldsymbol{y}_l)^2$$

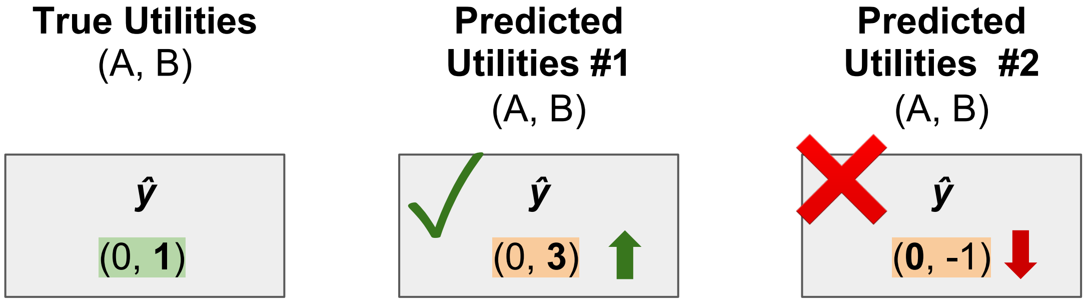
Loss Function Families (Step 2)

- (Approach 2) **“Directed Weighted-MSE”:**
  - Hypothesis: Over-predicting and under-predicting can have different consequences.



Loss Function Families (Step 2)

- (Approach 2) **"Directed Weighted-MSE":**
  - Hypothesis: Over-predicting and under-predicting can have different consequences.



  - Idea: Learn different parameters for over- and under-predicting

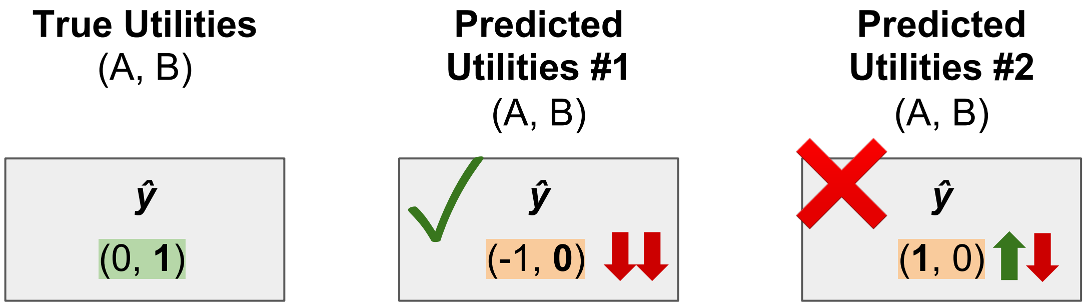
Loss Function Families (Step 2)

- (Approach 3) **"Quadratic"**:
  - Hypothesis: It's not just about whether individual predictions are over- or under-predict

True Utilities (A, B)	**Predicted Utilities #1** (A, B)	**Predicted Utilities #2** (A, B)
$\hat{y}$	$\hat{y}$	$\hat{y}$
(0, **1**)	(-1, **0**) ↓↓	(**1**, 0) ↑↓

Loss Function Families (Step 2)

- (Approach 3) **“Quadratic”:**
  - Hypothesis: It’s not just about whether individual predictions are over- or under-predict



  - Idea: Learn a low-rank symmetric PSD matrix *H*

$$(\hat{\boldsymbol{y}} - \boldsymbol{y})^T H (\hat{\boldsymbol{y}} - \boldsymbol{y})$$

Loss Function Families (Step 2)

- (Approach 3) "Quadratic":
  - Alternate Interpretation: Equals 2nd-order Taylor series approximation of ***DL*** at ***ŷ = y***

$$
\begin{aligned}
DL(\overbrace{\boldsymbol{y}_n + \boldsymbol{\epsilon}}^{\hat{\boldsymbol{y}}_n}, \boldsymbol{y}_n) &= \overbrace{DL(\boldsymbol{y}_n, \boldsymbol{y}_n)}^{\text{constant}} + \overbrace{\nabla_{\hat{\boldsymbol{y}}_n} DL(\boldsymbol{y}_n, \boldsymbol{y}_n)}^{0 \leftarrow (\boldsymbol{y}_n, \boldsymbol{y}_n) \text{ is a minima}} \boldsymbol{\epsilon} \\
&\quad + \boldsymbol{\epsilon}^T \underbrace{\nabla^2_{\hat{\boldsymbol{y}}_n} DL(\boldsymbol{y}_n, \boldsymbol{y}_n)}_{\text{Hessian } H} \boldsymbol{\epsilon} + \ldots \\
&\approx DL(\boldsymbol{y}_n, \boldsymbol{y}_n) + (\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n)^T H (\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n)
\end{aligned}
$$

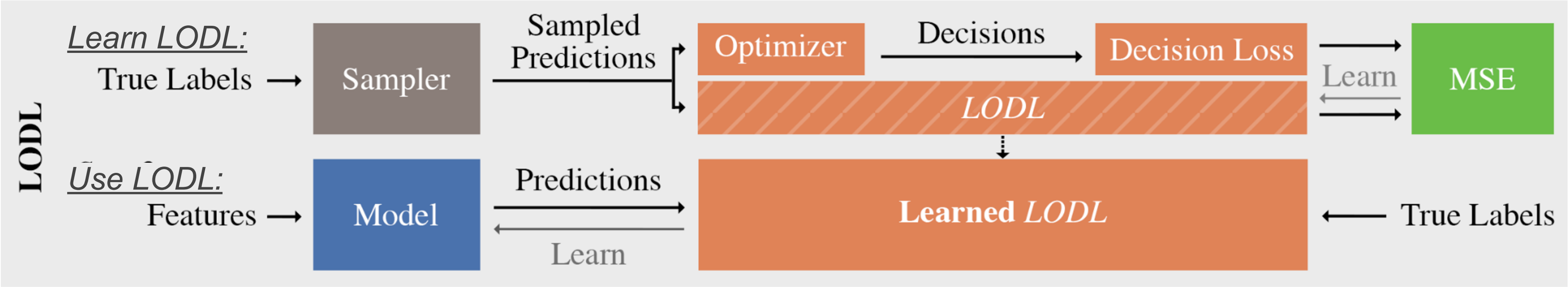
Loss Function Families (Step 2)

- (Approach 3) "Quadratic":
  - Alternate Interpretation: Equals 2nd-order Taylor series approximation of ***DL*** at ***ŷ = y***

$$\begin{aligned} DL(\overbrace{\boldsymbol{y}_n + \boldsymbol{\epsilon}}^{\hat{\boldsymbol{y}}_n}, \boldsymbol{y}_n) &= \overbrace{DL(\boldsymbol{y}_n, \boldsymbol{y}_n)}^{\text{constant}} + \overbrace{\nabla_{\hat{\boldsymbol{y}}_n} DL(\boldsymbol{y}_n, \boldsymbol{y}_n)}^{0 \leftarrow (\boldsymbol{y}_n, \boldsymbol{y}_n) \text{ is a minima}} \boldsymbol{\epsilon} \\ &\quad + \boldsymbol{\epsilon}^T \underbrace{\nabla^2_{\hat{\boldsymbol{y}}_n} DL(\boldsymbol{y}_n, \boldsymbol{y}_n)}_{\text{Hessian } H} \boldsymbol{\epsilon} + \ldots \\ &\approx DL(\boldsymbol{y}_n, \boldsymbol{y}_n) + (\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n)^T H (\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n) \end{aligned}$$

$$\implies DL(\hat{\boldsymbol{y}}_n, \boldsymbol{y}_n) - DL(\boldsymbol{y}_n, \boldsymbol{y}_n) \approx (\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n)^T H (\hat{\boldsymbol{y}}_n - \boldsymbol{y}_n)$$

Overall Approach



Outline

- Introduction
- Predict-Then-Optimize Details
- Our Approach
- **Experiments**
- Conclusions and Future Work

Domains

Three resource allocation domains from the literature:

1. **Linear Model:** Top-K
   - predictive model is linear, but underlying distribution is cubic
2. **Web Advertising:** Submodular Maximization
   - Predict CTRs, decide which websites on which to advertise
3. **Portfolio Optimization:** Quadratic Program
   - Predict future stock value, maximize “return” - “risk”

Baselines

- Upper and Lower Bounds:
  - **Random:** Randomly sample a value from *U[0, 1]*
  - **Optimal:** Use true parameters as predictions

- Past Approaches:
  - **2-Stage (MSE):** Train predictive model with MSE
  - **DFL:** Using the surrogate from the literature

- Importance of Convexity:
  - **NN-based “Decision Loss”**

Results 1: Performance on 3 Domains

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**
DirectedQuadratic	**0.962 ± 0.000**	0.910 ± 0.043	0.325 ± 0.014

Takeaway 1: Directed Quadratic does well consistently *without handcrafting!*

Results 1: Performance on 3 Domains

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

Takeaway 2: Lack of Convexity can lead to inconsistent results

Results 1: Performance on 3 Domains

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

Takeaway 3: DFL has high variance (when surrogates are non-convex)

Results 2: Ablations (on Web Advertising domain)

Approach	Normalized Test DQ (1-Perturbed)	Normalized Test DQ (2-Perturbed)	Normalized Test DQ (All-Perturbed)
NN	0.855 ± 0.121	$\mathbf{0.888 \pm 0.086}$	0.802 ± 0.159
WeightedMSE	0.496 ± 0.138	0.533 ± 0.139	$\mathbf{0.576 \pm 0.151}$
DirectedWeightedMSE	0.470 ± 0.150	$\mathbf{0.533 \pm 0.160}$	0.500 ± 0.130
Quadratic	0.773 ± 0.250	0.877 ± 0.097	$\mathbf{0.918 \pm 0.048}$
DirectedQuadratic	0.770 ± 0.187	0.842 ± 0.109	$\mathbf{0.845 \pm 0.080}$

Varying Sampling Strategy: Best strategy is dependent on the loss function family

Results 2: Ablations (on Web Advertising domain)

Approach	Normalized Test DQ (50 samples)	Normalized Test DQ (500 samples)	Normalized Test DQ (5000 samples)
NN	0.805 ± 0.134	0.802 ± 0.159	$\mathbf{0.814 \pm 0.137}$
WeightedMSE	0.496 ± 0.138	0.496 ± 0.139	$\mathbf{0.533 \pm 0.137}$
DirectedWeightedMSE	0.477 ± 0.147	0.533 ± 0.159	$\mathbf{0.533 \pm 0.149}$
Quadratic	0.677 ± 0.173	0.918 ± 0.048	$\mathbf{0.931 \pm 0.040}$
DirectedQuadratic	0.594 ± 0.134	0.845 ± 0.081	$\mathbf{0.910 \pm 0.043}$

Varying Number of Samples: More samples is better

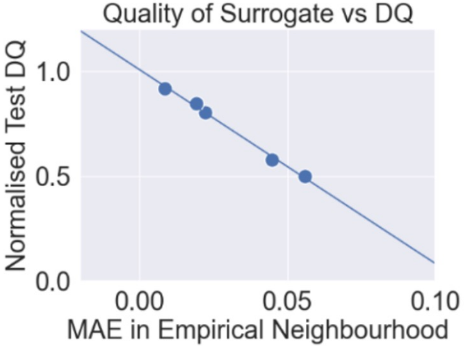
Results 3: Quality of Learned Loss vs. Decision Quality

- “Error” depends on distribution of interest:
  - **“Empirical Neighbourhood”:** True “predicted” parameters encountered while training predictive model
  - **“Gaussian Neighbourhood”:** Proxy for above calculated by adding noise to the “true” labels

- The second is an approximation for the first *(via the localness assumption)*

Results 3: Quality of Learned Loss vs. Decision Quality

Approach	MAE in Gaussian Neighborhood	MAE in Empirical Neighborhood	Normalized DQ on Test Data
NN	0.0094 ± 0.0006	0.0222 ± 0.0173	0.802 ± 0.159
WeightedMSE	0.0104 ± 0.0000	0.0448 ± 0.0171	0.576 ± 0.151
DirectedWeightedMSE	$\mathbf{0.0092 \pm 0.0000}$	0.0558 ± 0.0164	0.500 ± 0.130
Quadratic	0.0096 ± 0.0000	$\mathbf{0.0086 \pm 0.0052}$	$\mathbf{0.918 \pm 0.048}$
DirectedQuadratic	0.0106 ± 0.0000	0.0191 ± 0.0079	0.845 ± 0.080



Decision Quality is correlated with Error in the Empirical Neighbourhood but *not* the Gaussian Neighbourhood!

Outline

- Introduction
- Predict-Then-Optimize Details
- Our Approach
- Experiments
- **Conclusions and Future Work**

Conclusions

- We provide a novel way to address the `non-differentiability’ of optimization problems in the context of predict-then-optimize

- We show that our approach performs well on 3 domains from the literature

Future Work: 1-Year Scale

- **Better Proxy for "Empirical Neighbourhood":** We see that the Gaussian Neighbourhood is not an ideal proxy.
  - Perhaps we can use a 2-stage model to sample points?

- **Theoretical Analysis of DFL:** So far, we only show that our approach outperforms 2-stage on 3 domains. However, the results can be sensitive to small changes in the domain.
  - Can we analyze necessary conditions for the improvement?

Future Work: PhD Scale

- Better understand the **mechanism** behind why DFL does better than 2-stage and see if we can generalize that without DFL
    - More broadly, see if better "losses" improve ML outputs?
- Find a **real-world application** in which we can do better by incorporating task structure while learning

Acknowledgement

Kai Wang

Prof. Bryan Wilder

Prof. Andrew Perrault

Prof. Milind Tambe

Thank You!

Bibliography

1. Donti, Priya, Brandon Amos, and J. Zico Kolter. "Task-based end-to-end model learning in stochastic optimization." Advances in neural information processing systems 30 (2017).
2. Elmachtoub, Adam N., and Paul Grigas. "Smart “predict, then optimize”." Management Science 68.1 (2022): 9-26.
3. Wilder, Bryan, Bistra Dilkina, and Milind Tambe. "Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. No. 01. 2019.

Training Schematic