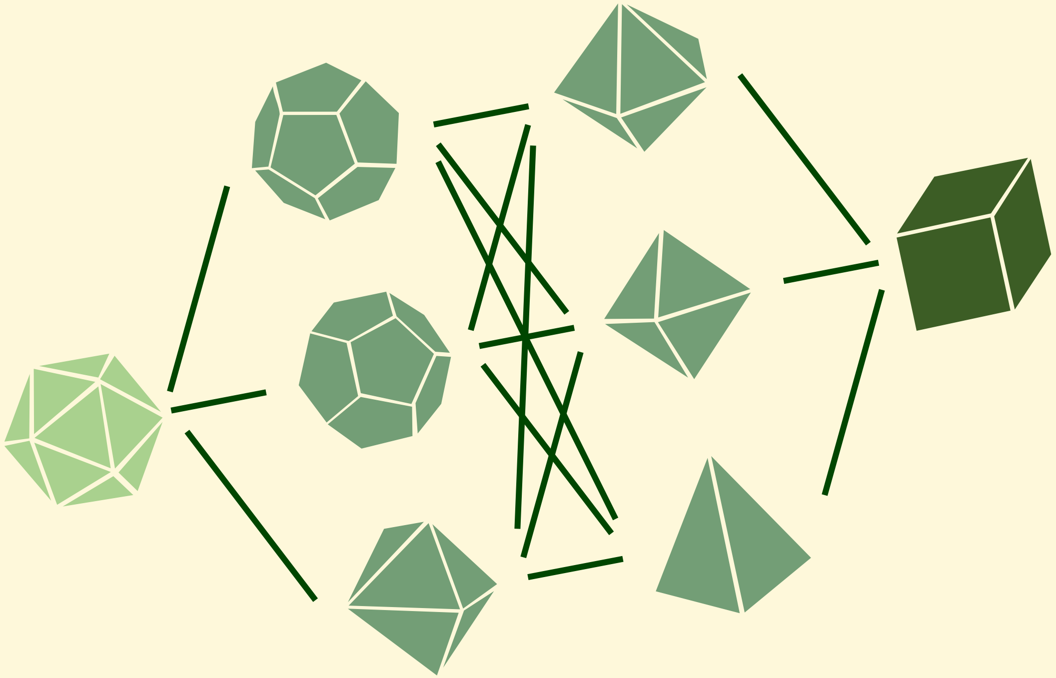
ACC 2024 | Physics-Informed ML for Modeling, Control and Optimization

PyEPO

A PyTorch-based End-to-End Predict-then-Optimize Library



Mechanical & Industrial Engineering
UNIVERSITY OF TORONTO

Presented by Bo Tang

July 9, 2024

Authors

Bo Tang

PhD Candidate
Department of Mechanical & Industrial Engineering, University of Toronto

Elias B. Khalil

Assistant Professor
Department of Mechanical & Industrial Engineering, University of Toronto

SCALE AI Research Chair
Data-Driven Algorithms for Modern Supply Chains

Linear Objective Function

$$\min_{\underbrace{\boldsymbol{w}}_{\text{Decision Variables}}} \{ \overbrace{\boldsymbol{c}^\top \boldsymbol{w}}^{\text{Linear Objective Function}} : \underbrace{\boldsymbol{w} \in \boldsymbol{S}}_{\text{Feasible Region}} \}$$

Use appropriate algorithm (MILP, MIQCP, CP, custom algorithms...) to obtain optimal solution $\boldsymbol{w}^*(\boldsymbol{c})$

Unknown Cost Coefficients

$$\min_{\boldsymbol{w}}\{\boldsymbol{c}_1{}^\top \boldsymbol{w}: \boldsymbol{w} \in \boldsymbol{S}\}$$

$$\min_{\boldsymbol{w}}\{\boldsymbol{c}_2{}^\top \boldsymbol{w}: \boldsymbol{w} \in \boldsymbol{S}\}$$

$$\min_{\boldsymbol{w}}\{\boldsymbol{c}_3{}^\top \boldsymbol{w}: \boldsymbol{w} \in \boldsymbol{S}\}$$

$$\ddots$$

Unknown Cost Coefficients

Unknown Cost

$$\min_{\boldsymbol{w}} \{\boldsymbol{c}_1^\top \boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S}\}$$

$$\min_{\boldsymbol{w}} \{\boldsymbol{c}_2^\top \boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S}\}$$

$$\min_{\boldsymbol{w}} \{\boldsymbol{c}_3^\top \boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S}\}$$

$$\ddots$$

Identical Constraints

Unknown Cost Coefficients

Unknown Cost

$$\min_{\boldsymbol{w}} \{\boldsymbol{c}_1{}^\top \boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S}\}$$

$$\min_{\boldsymbol{w}} \{\boldsymbol{c}_2{}^\top \boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S}\}$$

$$\min_{\boldsymbol{w}} \{\boldsymbol{c}_3{}^\top \boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S}\}$$

$\ddots$

Identical Constraints

Observed Feature Vector

$$\boldsymbol{x}_1$$

$$\boldsymbol{x}_2$$

$$\boldsymbol{x}_3$$

$$\vdots$$

Unknown Cost Coefficients

Unknown Cost

$$\min_{\boldsymbol{w}} \{ \boldsymbol{c}_1^\top \boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S} \}$$

$$\min_{\boldsymbol{w}} \{ \boldsymbol{c}_2^\top \boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S} \}$$

$$\min_{\boldsymbol{w}} \{ \boldsymbol{c}_3^\top \boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S} \}$$

$$\ddots$$

Identical Constraints

$$\hat{\boldsymbol{c}} = g(\boldsymbol{x}; \boldsymbol{\theta})$$

Prediction

Observed Feature Vector

$$\boldsymbol{x}_1$$

$$\boldsymbol{x}_2$$

$$\boldsymbol{x}_3$$

$$\vdots$$

Unknown Cost Coefficients

Unknown Cost

$$\min_{\boldsymbol{w}} \{\boldsymbol{c}_1{}^{\top} \boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S}\}$$

$$\min_{\boldsymbol{w}} \{\boldsymbol{c}_2{}^{\top} \boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S}\}$$

$$\min_{\boldsymbol{w}} \{\boldsymbol{c}_3{}^{\top} \boldsymbol{w} : \boldsymbol{w} \in \boldsymbol{S}\}$$

$\ddots$

Identical Constraints

Predicted Costs

Model Parameters

$$\hat{\boldsymbol{c}} = g(\boldsymbol{x}; \boldsymbol{\theta})$$

Prediction

Observed Feature Vector

$\boldsymbol{x}_1$

$\boldsymbol{x}_2$

$\boldsymbol{x}_3$

$\vdots$

Examples

- Vehicle Routing
- Energy Scheduling
- Portfolio Optimization

Examples

- **Vehicle Routing**
- **Energy Scheduling**
- **Portfolio Optimization**

Unknown Costs: Travel Time, Electricity Prices, Asset Returns

Examples

❖ **Vehicle Routing**

❖ **Energy Scheduling**

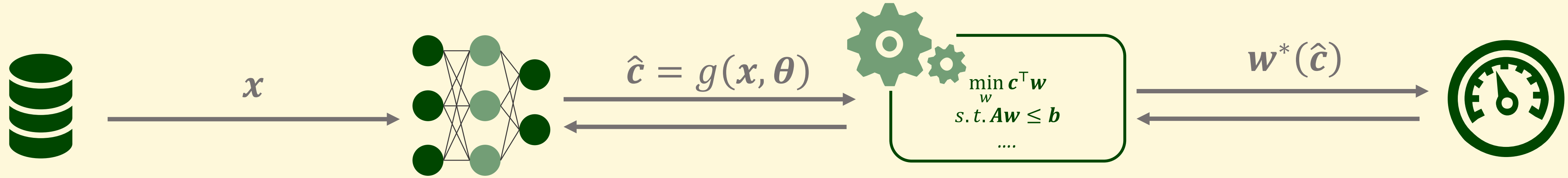
❖ **Portfolio Optimization**

Unknown Costs: Travel Time, Electricity Prices, Asset Returns

⬆

Observed Features: Distance, Time, Weather, Financial Factors…

End-to-End Predict-then-Optimize



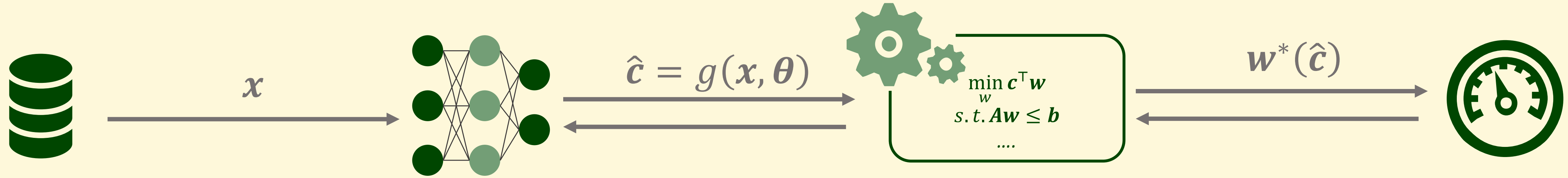
dataset $\mathcal{D}$ with data points $(\boldsymbol{x}, \boldsymbol{c})$

Prediction model $g(\boldsymbol{x}, \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$

optimization solver $\boldsymbol{w}^*(\hat{\boldsymbol{c}}) = \operatorname{argmin}_{\boldsymbol{w} \epsilon S} \hat{\boldsymbol{c}}^\top \boldsymbol{w}$

loss function $\mathcal{L}(\cdot)$ to measure decision error

End-to-End Predict-then-Optimize



dataset $\mathcal{D}$ with data points (x, c)

Prediction model $g(x, \theta)$ with parameters θ

optimization solver $w^*(\hat{c}) = \operatorname{argmin}_{w \in S} \hat{c}^\top w$

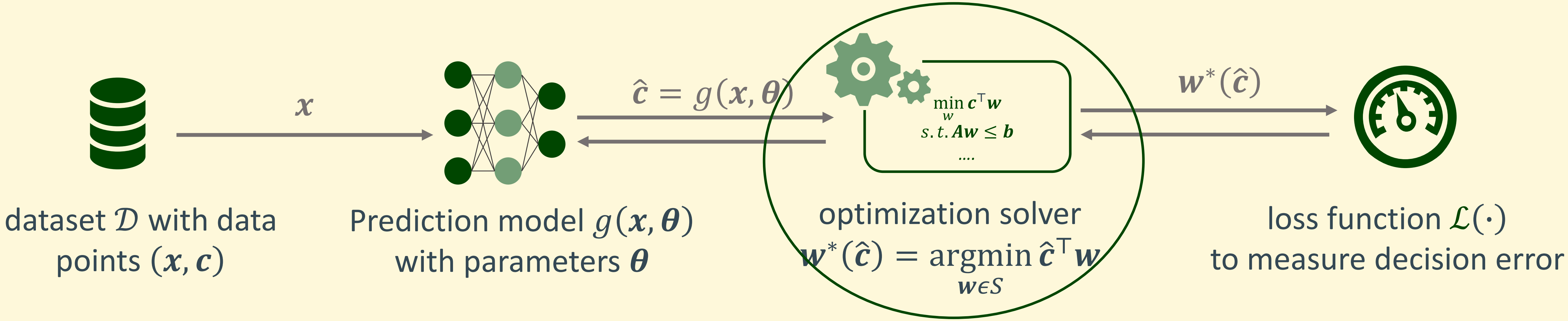
loss function $\mathcal{L}(\cdot)$ to measure decision error

e.g.

$$\mathcal{L}_{\text{Regret}}(\hat{c}, c) = \overbrace{c^\top w^*(\hat{c})}^{\text{Optimal solution if you optimize with } \hat{c}} - \overbrace{c^\top w^*(c)}^{\text{Optimal value with true cost } c}$$

$$\mathcal{L}_{\text{Square}}(\hat{c}, c) = \frac{1}{2} \| w^*(c) - w^*(\hat{c}) \|_2^2$$

End-to-End Predict-then-Optimize



Algorithm 1 End-to-end Learning

Require: coefficient matrix $\boldsymbol{A}$, right-hand side $\boldsymbol{b}$, data $\mathcal{D}$

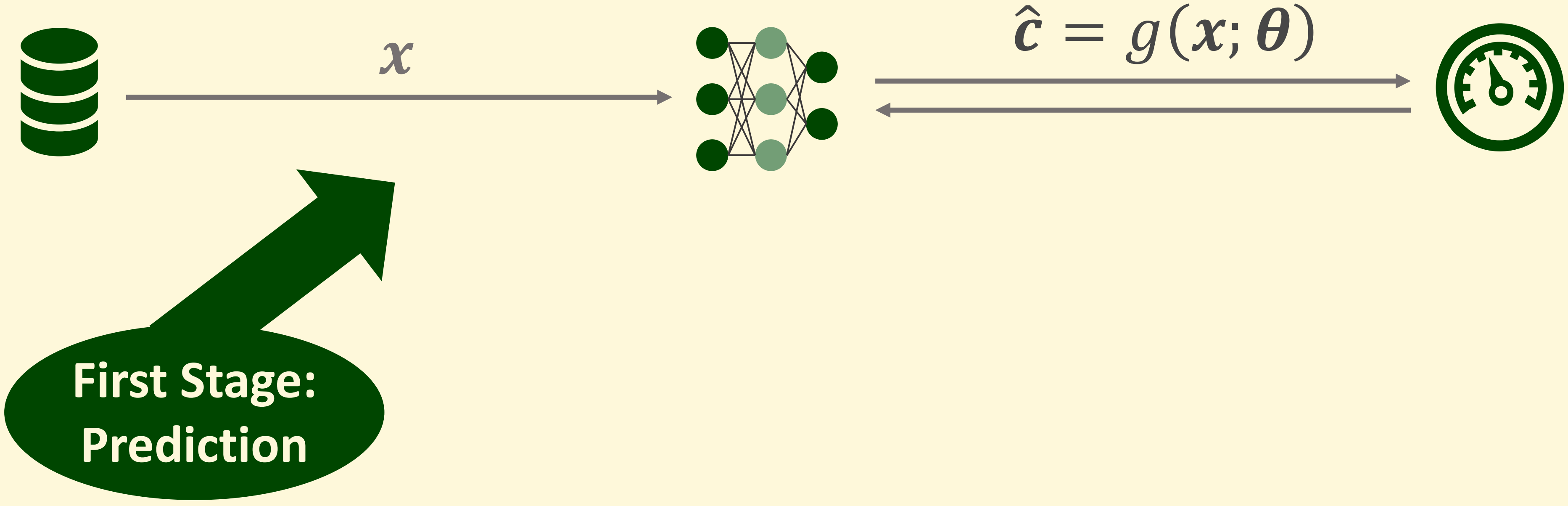
```
Initialize predictor parameters θ for predictor g(x; θ)
for epochs do
    for each batch of training data (x, c) do
        Sample batch of the cost vectors c with the corresponding features x
        Predict cost using predictor ĉ := g(x; θ)
        Forward pass to compute optimal solution w*_ĉ := argmin_{w∈S} ĉ^T w
        Forward pass to compute decision loss l(·)
        Backward pass from loss l(·) to update parameters θ with gradient
    end for
end for
```

Two-Stage Predict-then-Optimize

Dataset $\mathcal{D}$
with data points $(\boldsymbol{x}, \boldsymbol{c})$

Prediction model $g(\boldsymbol{x}; \boldsymbol{\theta})$
with parameters $\boldsymbol{\theta}$

Loss function $\mathcal{L}(\cdot)$
to measure prediction error

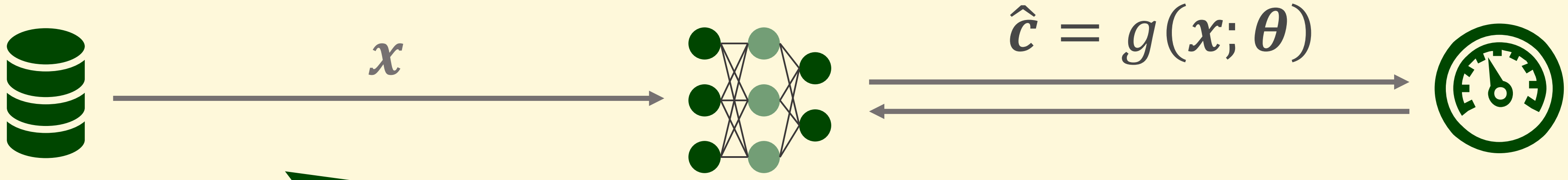


Two-Stage Predict-then-Optimize

Dataset $\mathcal{D}$ with data points $(\boldsymbol{x}, \boldsymbol{c})$

Prediction model $g(\boldsymbol{x}; \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$

Loss function $\mathcal{L}(\cdot)$ to measure prediction error



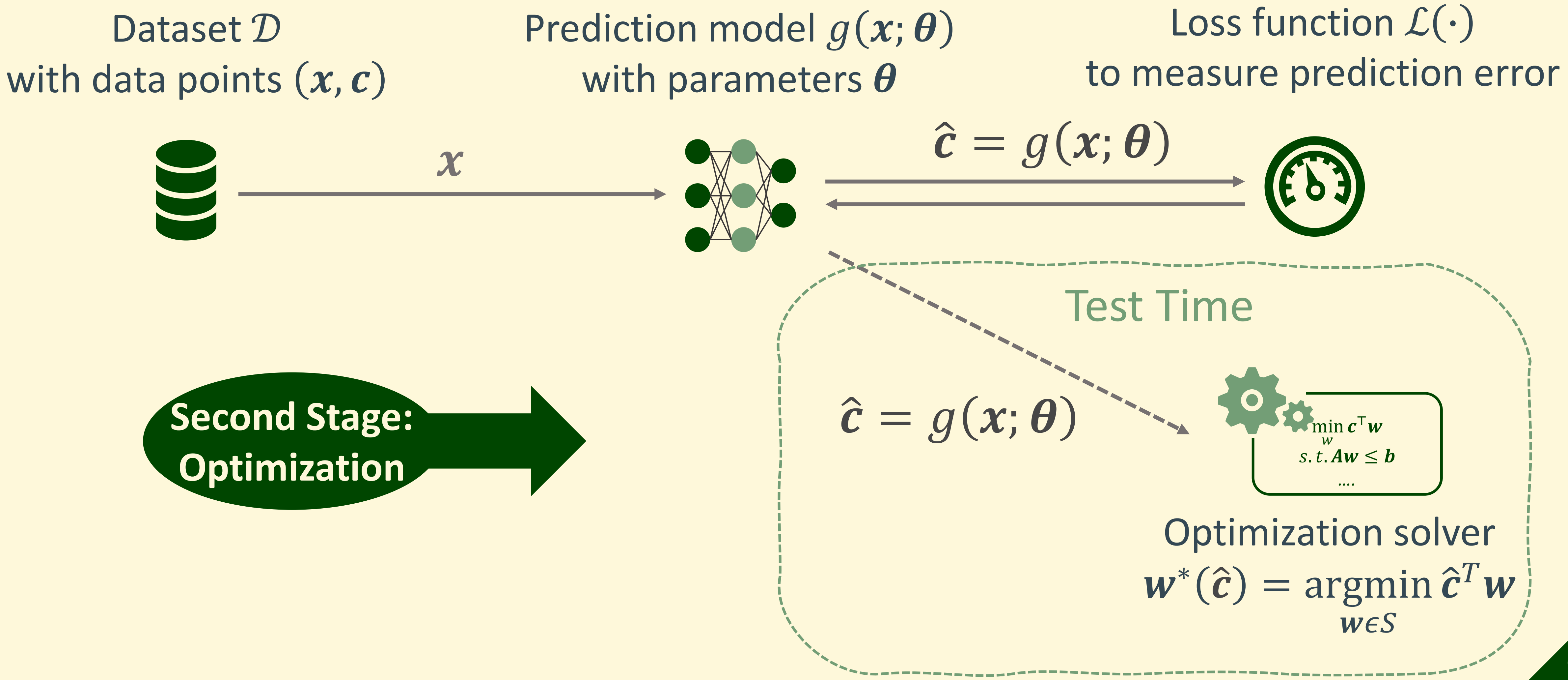
First Stage: Prediction

e.g.

$$\mathcal{L}_{\mathrm{MSE}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = \frac{1}{2} \|\boldsymbol{c} - \hat{\boldsymbol{c}}\|_2^2$$

$$\mathcal{L}_{\mathrm{MAE}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = \|\boldsymbol{c} - \hat{\boldsymbol{c}}\|_1$$

Two-Stage Predict-then-Optimize



Mismatch of Prediction and Decision Error

For example:

$$\max_{w_1,w_2} c_1 w_1 + c_2 w_2$$
$$s.t. \quad w_1 + w_2 \leq 1$$
$$w_1, w_2 \geq 0$$

Assume the true cost is $\boldsymbol{c} = (0,1)$, the optimal solution is $\boldsymbol{w}^*(\boldsymbol{c}) = (0,1)$

If the prediction $\hat{\boldsymbol{c}} = (1,0)$, the solution $\boldsymbol{w}^*(\hat{\boldsymbol{c}}) =$ and $\mathcal{L}_{\text{MSE}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) =$;

If the prediction $\hat{\boldsymbol{c}} = (0,3)$, the solution $\boldsymbol{w}^*(\hat{\boldsymbol{c}}) =$ and $\mathcal{L}_{\text{MSE}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) =$

Mismatch of Prediction and Decision Error

For example:

$$\max_{w_1,w_2} c_1 w_1 + c_2 w_2$$
$$s.t. \quad w_1 + w_2 \leq 1$$
$$w_1, w_2 \geq 0$$

Assume the true cost is $\boldsymbol{c} = (0,1)$, the optimal solution is $\boldsymbol{w}^*(\boldsymbol{c}) = (0,1)$
If the prediction $\hat{\boldsymbol{c}} = (1,0)$, the solution $\boldsymbol{w}^*(\hat{\boldsymbol{c}}) = (1,0)$ and $\mathcal{L}_{\mathrm{MSE}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = 1$;
If the prediction $\hat{\boldsymbol{c}} = (0,3)$, the solution $\boldsymbol{w}^*(\hat{\boldsymbol{c}}) = (0,1)$ and $\mathcal{L}_{\mathrm{MSE}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = 2$

Mismatch of Prediction and Decision Error

For example:

$$\max_{w_1, w_2} c_1 w_1 + c_2 w_2$$
$$s.t. \quad w_1 + w_2 \leq 1$$
$$w_1, w_2 \geq 0$$

Assume the true cost is $\boldsymbol{c} = (0,1)$, the optimal solution is $\boldsymbol{w}^*(\boldsymbol{c}) = (0,1)$

If the prediction $\hat{\boldsymbol{c}} = (1,0)$, the solution $\boldsymbol{w}^*(\hat{\boldsymbol{c}}) = (1,0)$ and $\mathcal{L}_{\text{MSE}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = 1$;

If the prediction $\hat{\boldsymbol{c}} = (0,3)$, the solution $\boldsymbol{w}^*(\hat{\boldsymbol{c}}) = (0,1)$ and $\mathcal{L}_{\text{MSE}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = 2$

Prediction error such as $l_{\text{MSE}}(\hat{c}, c)$ cannot measure the quality of decision.

Mismatch of Prediction and Decision Error

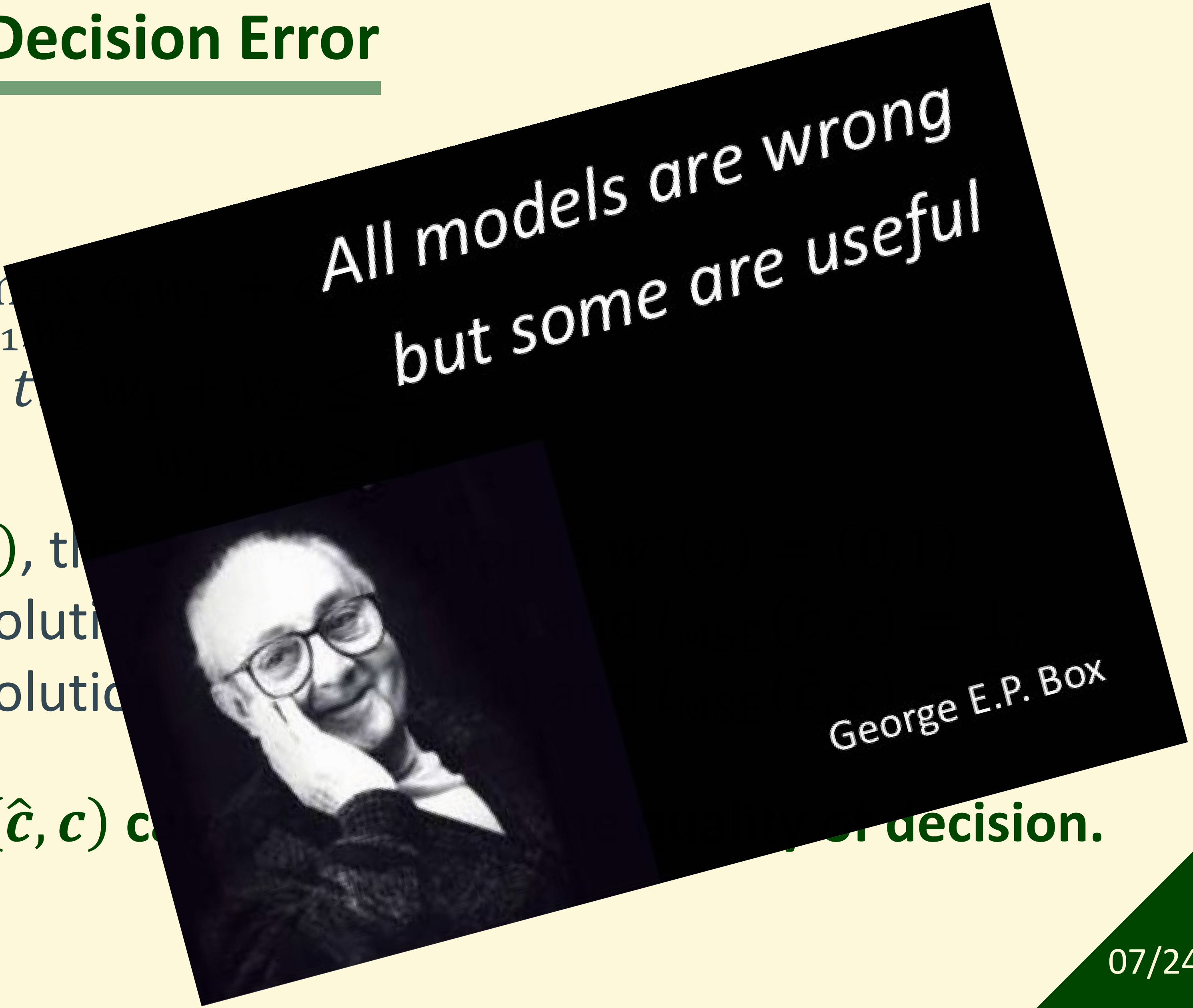
For example:

$$\mathrm{m}\ldots_{w_1 \ldots}$$
$$s.t \ldots$$

Assume the true cost is $\boldsymbol{c} = (0,1)$, th

If the prediction $\hat{\boldsymbol{c}} = (1,0)$, the soluti

If the prediction $\hat{\boldsymbol{c}} = (0,3)$, the solutio



Prediction error such as $l_{\mathrm{MSE}}(\hat{\boldsymbol{c}}, \boldsymbol{c})$ ca ... of decision.

Imitation Learning and Parametric Optimization

Directly predict: $\hat{\boldsymbol{w}}^* = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta})$

- Let $\hat{\boldsymbol{w}}^*$ close to the true optimal $\boldsymbol{w}^*$
- Reduce objective function $f(\hat{\boldsymbol{w}}^*)$

Imitation Learning and Parametric Optimization

Directly predict: $\hat{\boldsymbol{w}}^* = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta})$

- Let $\hat{\boldsymbol{w}}^*$ close to the true optimal $\boldsymbol{w}^*$
- Reduce objective function $f(\hat{\boldsymbol{w}}^*)$

Efficiency

Avoid the major bottleneck in computational efficiency: optimizing

Imitation Learning and Parametric Optimization

Directly predict: $\hat{\boldsymbol{w}}^* = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\theta})$

- Let $\hat{\boldsymbol{w}}^*$ close to the true optimal $\boldsymbol{w}^*$
- Reduce objective function $f(\hat{\boldsymbol{w}}^*)$

Efficiency

Avoid the major bottleneck in computational efficiency: optimizing

Feasibility

Prediction often faces feasibility issues

Chain-Rule of Gradient

Algorithm 1 End-to-end Learning

Require: coefficient matrix $\boldsymbol{A}$, right-hand side $\boldsymbol{b}$, data $\mathcal{D}$

```
Initialize predictor parameters θ for predictor g(x; θ)
for epochs do
    for each batch of training data (x, c) do
        Sample batch of the cost vectors c with the corresponding features x
        Predict cost using predictor ĉ := g(x; θ)
        Forward pass to compute optimal solution w*_ĉ := argmin_{w∈S} ĉᵀw
        Forward pass to compute decision loss l(·)
        Backward pass from loss l(·) to update parameters θ with gradient
    end for
end for
```

Chain Rule:

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \boldsymbol{\theta}} =$$

Chain-Rule of Gradient

Algorithm 1 End-to-end Learning

Require: coefficient matrix $\boldsymbol{A}$, right-hand side $\boldsymbol{b}$, data $\mathcal{D}$

1: Initialize predictor parameters $\boldsymbol{\theta}$ for predictor $g(\boldsymbol{x}; \boldsymbol{\theta})$
2: **for** epochs **do**
3:     **for** each batch of training data $(\boldsymbol{x}, \boldsymbol{c})$ **do**
4:         Sample batch of the cost vectors $\boldsymbol{c}$ with the corresponding features $\boldsymbol{x}$
5:         Predict cost using predictor $\hat{\mathbf{c}} := g(\boldsymbol{x}; \boldsymbol{\theta})$
6:         Forward pass to compute optimal solution $\boldsymbol{w}^*_{\hat{\mathbf{c}}} := \text{argmin}_{\boldsymbol{w} \in S} \hat{\mathbf{c}}^T \boldsymbol{w}$
7:         Forward pass to compute decision loss $l(\cdot)$
8:         Backward pass from loss $l(\cdot)$ to update parameters $\boldsymbol{\theta}$ with gradient
9:     **end for**
10: **end for**

Chain Rule:

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \boldsymbol{\theta}} = \frac{\partial \mathcal{L}(\cdot)}{\partial \hat{\mathbf{c}}} \frac{\partial \hat{\mathbf{c}}}{\partial \boldsymbol{\theta}}$$

Chain-Rule of Gradient

Algorithm 1 End-to-end Learning

Require: coefficient matrix $\boldsymbol{A}$, right-hand side $\boldsymbol{b}$, data $\mathcal{D}$

Initialize predictor parameters $\boldsymbol{\theta}$ for predictor $g(\boldsymbol{x}; \boldsymbol{\theta})$
for epochs **do**
  for each batch of training data $(\boldsymbol{x}, \boldsymbol{c})$ **do**
    Sample batch of the cost vectors $\boldsymbol{c}$ with the corresponding features $\boldsymbol{x}$
    Predict cost using predictor $\hat{\mathbf{c}} := g(\boldsymbol{x}; \boldsymbol{\theta})$
    Forward pass to compute optimal solution $\boldsymbol{w}^*_{\hat{\mathbf{c}}} := \text{argmin}_{\boldsymbol{w} \in S} \hat{\mathbf{c}}^T \boldsymbol{w}$
    Forward pass to compute decision loss $l(\cdot)$
    Backward pass from loss $l(\cdot)$ to update parameters $\boldsymbol{\theta}$ with gradient
  end for
end for

Chain Rule:

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \boldsymbol{\theta}} = \frac{\partial \mathcal{L}(\cdot)}{\partial \hat{\mathbf{c}}} \frac{\partial \hat{\mathbf{c}}}{\partial \boldsymbol{\theta}}$$

Easy:
Gradient of predicted costs to model parameters

Chain-Rule of Gradient

Algorithm 1 End-to-end Learning

Require: coefficient matrix $\boldsymbol{A}$, right-hand side $\boldsymbol{b}$, data $\mathcal{D}$

Initialize predictor parameters $\boldsymbol{\theta}$ for predictor $g(\boldsymbol{x}; \boldsymbol{\theta})$
for epochs **do**
  for each batch of training data $(\boldsymbol{x}, \boldsymbol{c})$ **do**
    Sample batch of the cost vectors $\boldsymbol{c}$ with the corresponding features $\boldsymbol{x}$
    Predict cost using predictor $\hat{\boldsymbol{c}} := g(\boldsymbol{x}; \boldsymbol{\theta})$
    Forward pass to compute optimal solution $\boldsymbol{w}^*_{\hat{\boldsymbol{c}}} := \text{argmin}_{\boldsymbol{w} \in S} \hat{\boldsymbol{c}}^T \boldsymbol{w}$
    Forward pass to compute decision loss $l(\cdot)$
    Backward pass from loss $l(\cdot)$ to update parameters $\boldsymbol{\theta}$ with gradient
  end for
end for

Chain Rule:

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \boldsymbol{\theta}} = \frac{\partial \mathcal{L}(\cdot)}{\partial \hat{\mathbf{c}}} \frac{\partial \hat{\mathbf{c}}}{\partial \boldsymbol{\theta}}$$

Hard:

Decision loss vary with the predicted costs

Chain-Rule of Gradient

Algorithm 1 End-to-end Learning

Require: coefficient matrix $\boldsymbol{A}$, right-hand side $\boldsymbol{b}$, data $\mathcal{D}$

```
Initialize predictor parameters θ for predictor g(x; θ)
for epochs do
    for each batch of training data (x, c) do
        Sample batch of the cost vectors c with the corresponding features x
        Predict cost using predictor ĉ := g(x; θ)
        Forward pass to compute optimal solution w*_ĉ := argmin_{w∈S} ĉ^T w
        Forward pass to compute decision loss l(·)
        Backward pass from loss l(·) to update parameters θ with gradient
    end for
end for
```

Chain Rule:

$$\frac{\partial \mathcal{L}(\cdot)}{\partial \boldsymbol{\theta}} = \frac{\partial \mathcal{L}(\cdot)}{\partial \hat{\boldsymbol{c}}} \frac{\partial \hat{\boldsymbol{c}}}{\partial \boldsymbol{\theta}}$$

Hard:

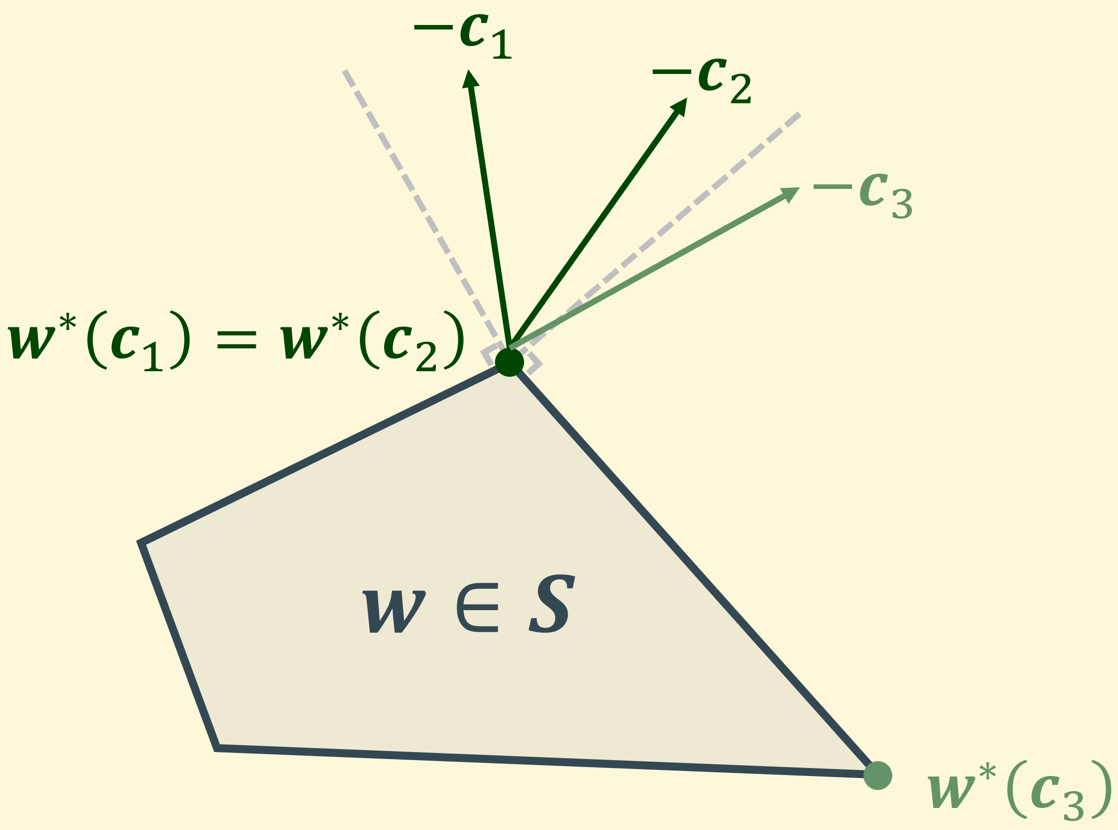
Decision loss vary with the predicted costs

$$\mathcal{L}_{\text{Regret}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = \boldsymbol{c}^{\top} \boldsymbol{w}^*(\hat{\boldsymbol{c}}) - \boldsymbol{c}^{\top} \boldsymbol{w}^*(\boldsymbol{c})$$

$$\mathcal{L}_{\text{Square}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = \frac{1}{2} \|\boldsymbol{w}^*(\boldsymbol{c}) - \boldsymbol{w}^*(\hat{\boldsymbol{c}})\|_2^2$$

with respect to $\boldsymbol{w}^*(\hat{\boldsymbol{c}})$!

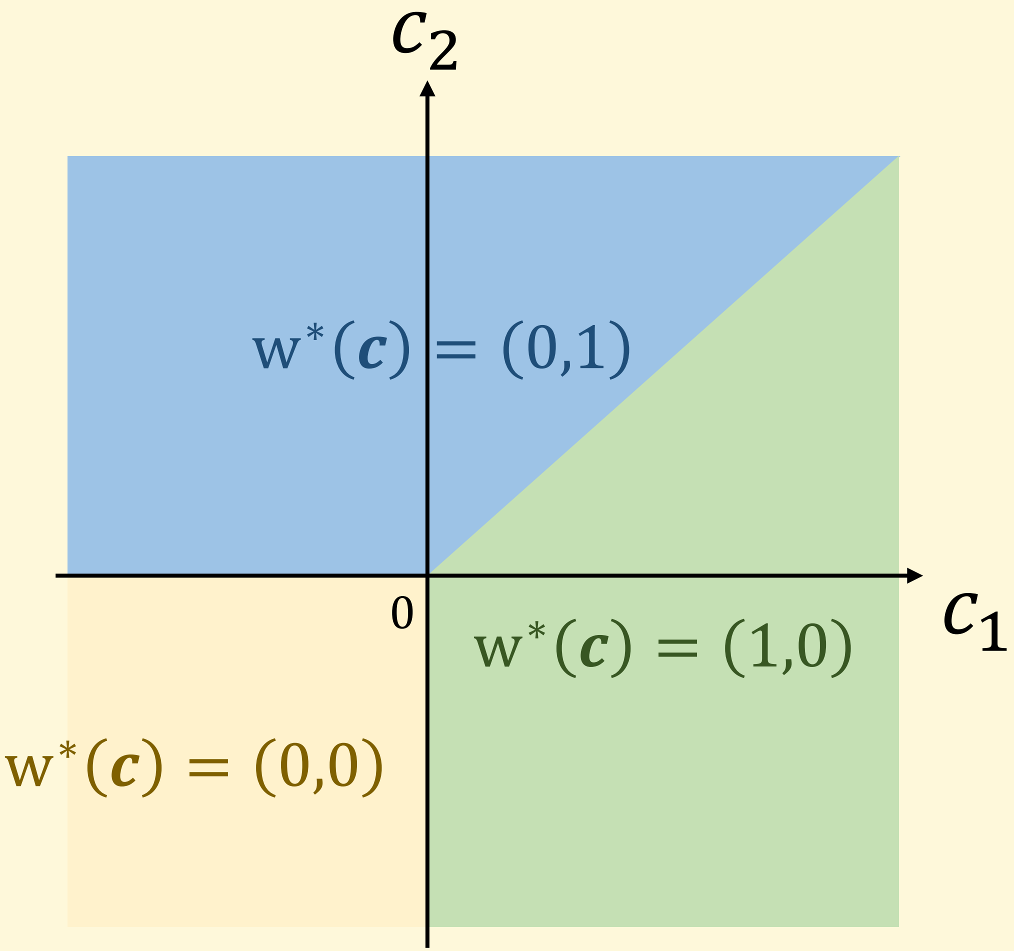
Piecewise Constant Function



$w^*(c)$ is a piecewise constant function!

Piecewise Constant Function

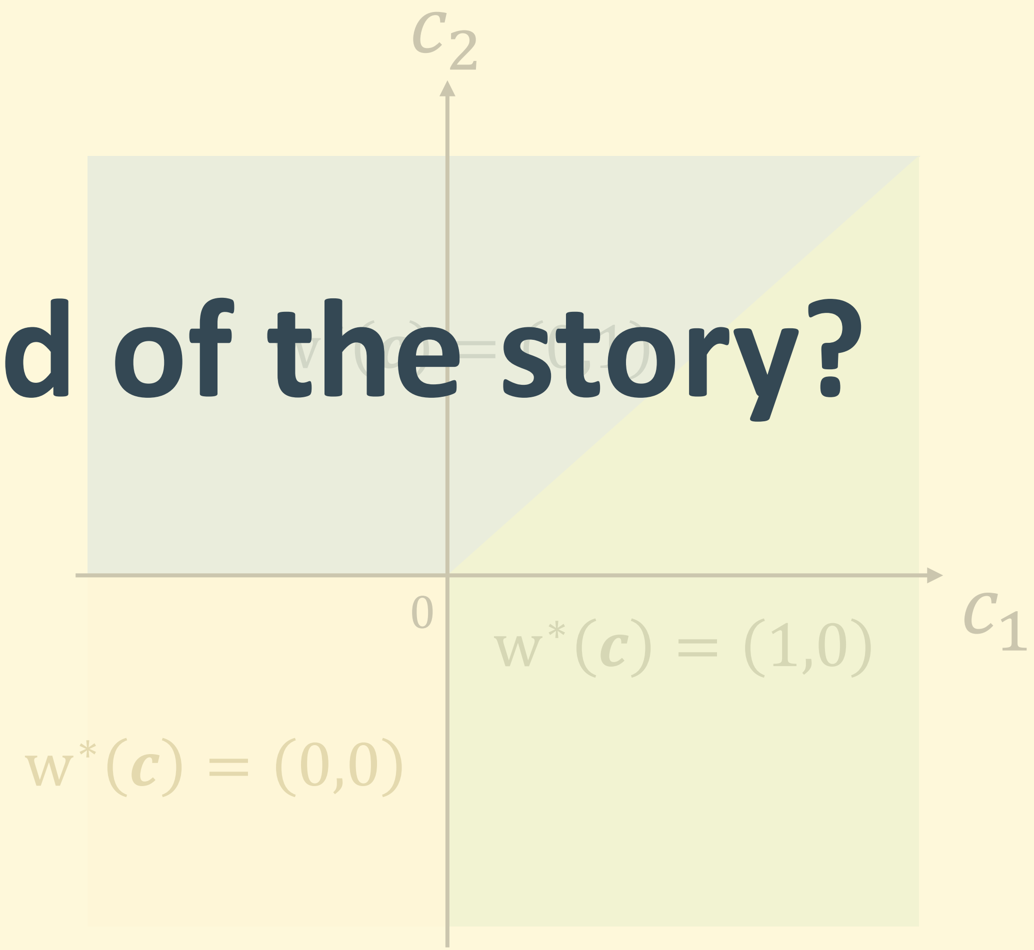
$$\max_{w_1, w_2} c_1 w_1 + c_2 w_2$$
$$s.t. \quad w_1 + w_2 \leq 1$$
$$w_1, w_2 \geq 0$$



Piecewise Constant Function

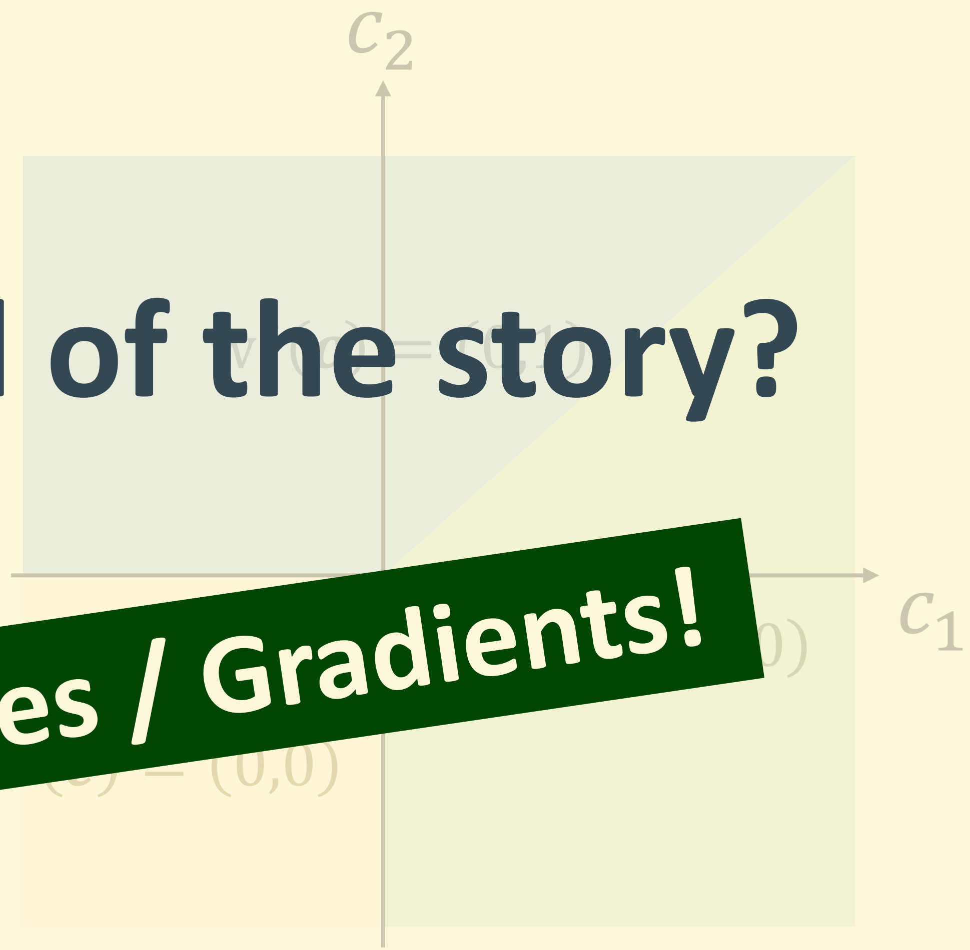
Is that the end of the story?

$$\max_{w_1,w_2} c_1 w_1 + c_2 w_2$$
$$s.t. \quad w_1 + w_2 \leq 1$$
$$w_1, w_2 \geq 0$$



Piecewise Constant Function

$$\max_{w_1,w_2} c_1 w_1 + c_2 w_2$$
$$s.t. \quad w_1 + w_2 \leq 1$$
$$w_1, w_2 \geq 0$$



Is that the end of the story?

Surrogate Losses / Gradients!

Derivative of Implicit Functions

OptNet:

- Solve the partial derivative matrix linear equations to calculate the solution and gradients for both **forward** and **backward** pass.
- Add a quadratic term to the linear objective function to obtain the **nonzero** gradient.

Karush-Kuhn-Tucker conditions

Given general problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{subject to} \quad & h_i(x) \leq 0, \quad i = 1, \dots m \\ & \ell_j(x) = 0, \quad j = 1, \dots r \end{aligned}$$

The **Karush-Kuhn-Tucker conditions** or **KKT conditions** are:

- $0 \in \partial f(x) + \sum_{i=1}^{m} u_i \partial h_i(x) + \sum_{j=1}^{r} v_j \partial \ell_j(x)$ (stationarity)
- $u_i \cdot h_i(x) = 0$ for all i (complementary slackness)
- $h_i(x) \leq 0, \ \ell_j(x) = 0$ for all i, j (primal feasibility)
- $u_i \geq 0$ for all i (dual feasibility)

- Amos, B., & Kolter, J. Z. (2017, July). Optnet: Differentiable optimization as a layer in neural networks. In International Conference on Machine Learning (pp. 136-145). PMLR.
- Wilder, B., Dilkina, B., & Tambe, M. (2019, July). Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, No. 01, pp. 1658-1665).

Smart "predict, then optimize"

A convex upper bound of $\mathcal{L}_{\mathrm{Regret}}(\hat{\boldsymbol{c}}, \boldsymbol{c})$:

- Elmachtoub, A. N., & Grigas, P. (2021). Smart "predict, then optimize". Management Science.

Smart "predict, then optimize"

A convex upper bound of $\mathcal{L}_{\mathrm{Regret}}(\hat{\boldsymbol{c}}, \boldsymbol{c})$:

$$\mathcal{L}_{\mathrm{SPO+}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = -\min_{\boldsymbol{w} \in \boldsymbol{W}} (2\hat{\boldsymbol{c}} - \boldsymbol{c})^\top \boldsymbol{w} + 2\hat{\boldsymbol{c}}^\top \boldsymbol{w}^*(\boldsymbol{c}) - \boldsymbol{c}^\top \boldsymbol{w}^*(\boldsymbol{c})$$

- Elmachtoub, A. N., & Grigas, P. (2021). Smart "predict, then optimize". Management Science.

Smart "predict, then optimize"

A convex upper bound of $\mathcal{L}_{\mathrm{Regret}}(\hat{\boldsymbol{c}}, \boldsymbol{c})$:

$$\mathcal{L}_{\mathrm{SPO+}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = -\underbrace{\min_{\boldsymbol{w} \in \boldsymbol{W}} (2\hat{\boldsymbol{c}} - \boldsymbol{c})^\top \boldsymbol{w}} + 2\hat{\boldsymbol{c}}^\top \boldsymbol{w}^*(\boldsymbol{c}) - \boldsymbol{c}^\top \boldsymbol{w}^*(\boldsymbol{c})$$

Computational overhead: We need to solve an optimization problem $\min_{\boldsymbol{w} \in \boldsymbol{W}} (2\hat{\boldsymbol{c}} - \boldsymbol{c})^\top \boldsymbol{w}$ per iteration.

• Elmachtoub, A. N., & Grigas, P. (2021). Smart "predict, then optimize". Management Science.

Smart "predict, then optimize"

A convex upper bound of $\mathcal{L}_{\text{Regret}}(\hat{\boldsymbol{c}}, \boldsymbol{c})$:

$$\mathcal{L}_{\text{SPO+}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = -\min_{\boldsymbol{w} \in \boldsymbol{W}} (2\hat{\boldsymbol{c}} - \boldsymbol{c})^\top \boldsymbol{w} + 2\hat{\boldsymbol{c}}^\top \boldsymbol{w}^*(\boldsymbol{c}) - \boldsymbol{c}^\top \boldsymbol{w}^*(\boldsymbol{c})$$

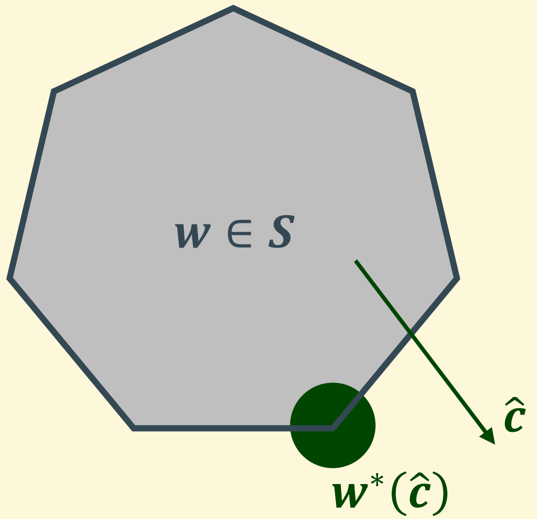
The subgradient of $\mathcal{L}_{\text{SPO+}}(\hat{\boldsymbol{c}}, \boldsymbol{c})$:

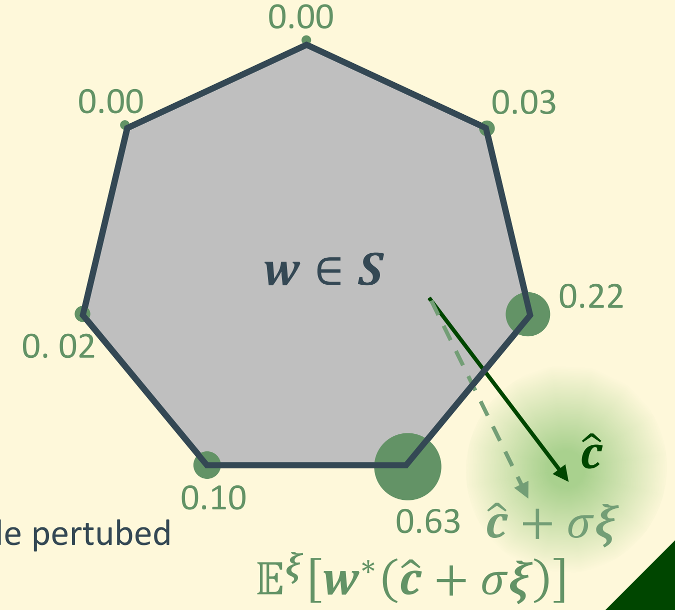
$$2\boldsymbol{w}^*(\boldsymbol{c}) - 2\boldsymbol{w}^*(2\hat{\boldsymbol{c}} - \boldsymbol{c}) \in \frac{\partial \mathcal{L}_{\text{SPO+}}(\hat{\boldsymbol{c}}, \boldsymbol{c})}{\partial \hat{\boldsymbol{c}}}$$

- Elmachtoub, A. N., & Grigas, P. (2021). Smart "predict, then optimize". Management Science.

Perturbed Method

Use random perturbation to deal with the cost vector $\hat{\boldsymbol{c}}$.



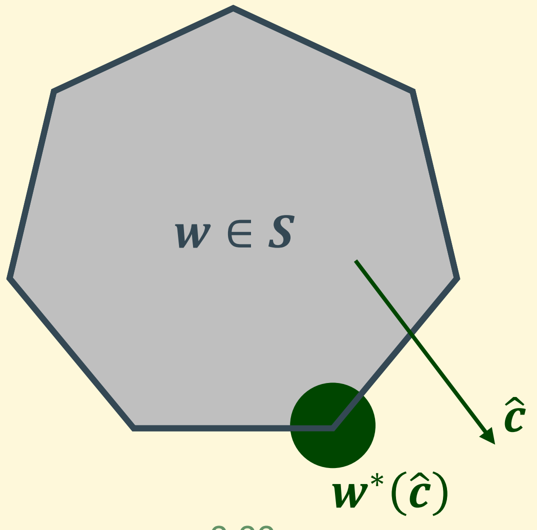


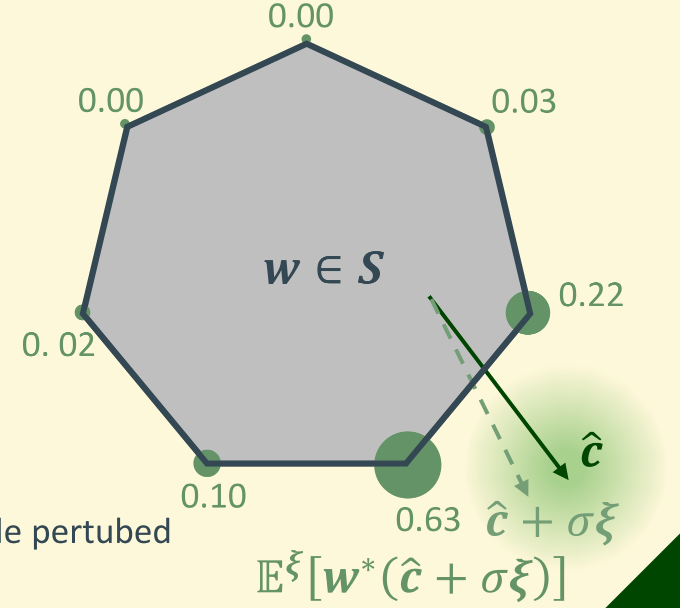
- Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J. P., & Bach, F. (2020). Learning with differentiable pertubed optimizers. Advances in neural information processing systems, 33, 9508-9519.
- Dalle, G., Baty, L., Bouvier, L., & Parmentier, A. (2022). Learning with combinatorial optimization layers: a probabilistic approach. arXiv preprint arXiv:2207.13513.c

Perturbed Method

Use random perturbation to deal with the cost vector $\hat{\boldsymbol{c}}$.

The expected value $\mathbb{E}^{\xi}[\boldsymbol{w}^*(\hat{\boldsymbol{c}} + \sigma\boldsymbol{\xi})]$ repalces $\boldsymbol{w}^*(\hat{\boldsymbol{c}})$, which is the weighted average (convex combination) of the extreme points of feasible region.



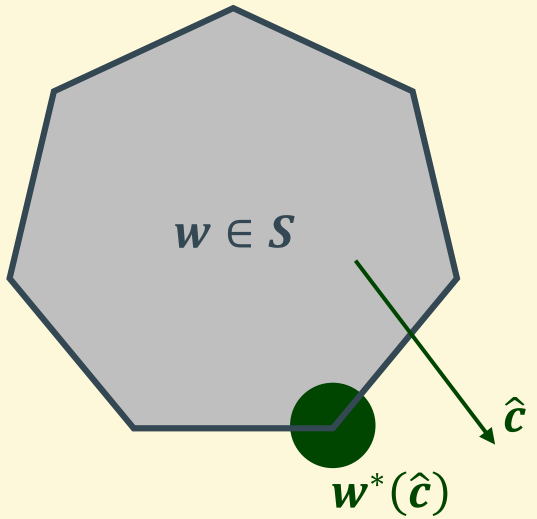


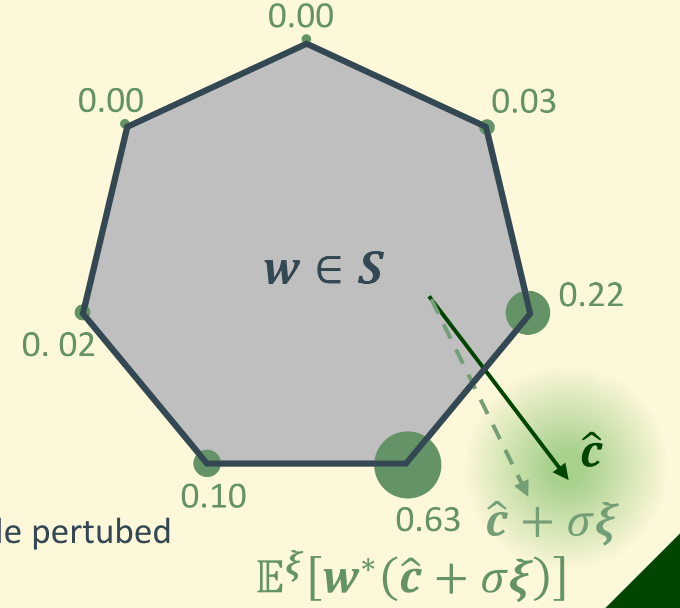
- Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J. P., & Bach, F. (2020). Learning with differentiable pertubed optimizers. Advances in neural information processing systems, 33, 9508-9519.
- Dalle, G., Baty, L., Bouvier, L., & Parmentier, A. (2022). Learning with combinatorial optimization layers: a probabilistic approach. arXiv preprint arXiv:2207.13513.c

Perturbed Method

Use random perturbation to deal with the cost vector $\hat{\boldsymbol{c}}$.

The expected value $\mathbb{E}^{\xi}[\boldsymbol{w}^*(\hat{\boldsymbol{c}} + \sigma \overbrace{\boldsymbol{\xi}}^{\text{A random perturbation } \boldsymbol{\xi} \sim \boldsymbol{N}(\boldsymbol{0}, \boldsymbol{1})})]$ repalces $\boldsymbol{w}^*(\hat{\boldsymbol{c}})$, which is the weighted average (convex combination) of the extreme points of feasible region.





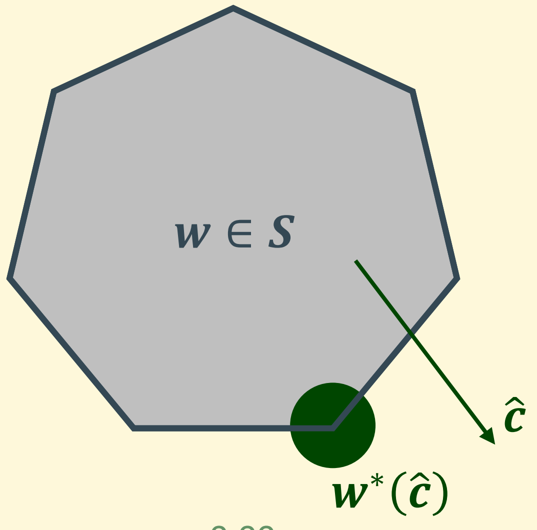
- Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J. P., & Bach, F. (2020). Learning with differentiable pertubed optimizers. Advances in neural information processing systems, 33, 9508-9519.
- Dalle, G., Baty, L., Bouvier, L., & Parmentier, A. (2022). Learning with combinatorial optimization layers: a probabilistic approach. arXiv preprint arXiv:2207.13513.c

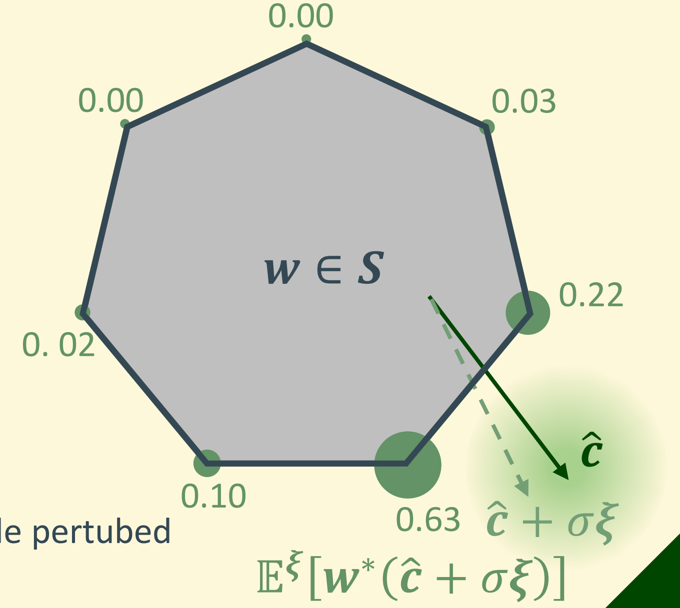
Perturbed Method

Use random perturbation to deal with the cost vector $\hat{\boldsymbol{c}}$.

The expected value $\mathbb{E}^{\xi}[\boldsymbol{w}^*(\hat{\boldsymbol{c}} + \sigma\boldsymbol{\xi})]$ repalces $\boldsymbol{w}^*(\hat{\boldsymbol{c}})$, which is the weighted average (convex combination) of the extreme points of feasible region.

How to calculate the expectation?





- Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J. P., & Bach, F. (2020). Learning with differentiable pertubed optimizers. Advances in neural information processing systems, 33, 9508-9519.
- Dalle, G., Baty, L., Bouvier, L., & Parmentier, A. (2022). Learning with combinatorial optimization layers: a probabilistic approach. arXiv preprint arXiv:2207.13513.c

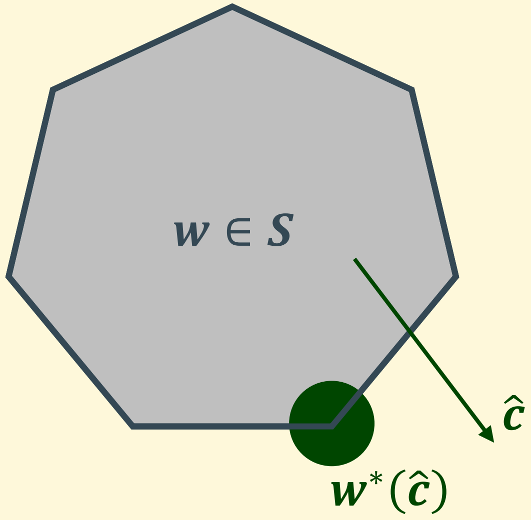
Perturbed Method

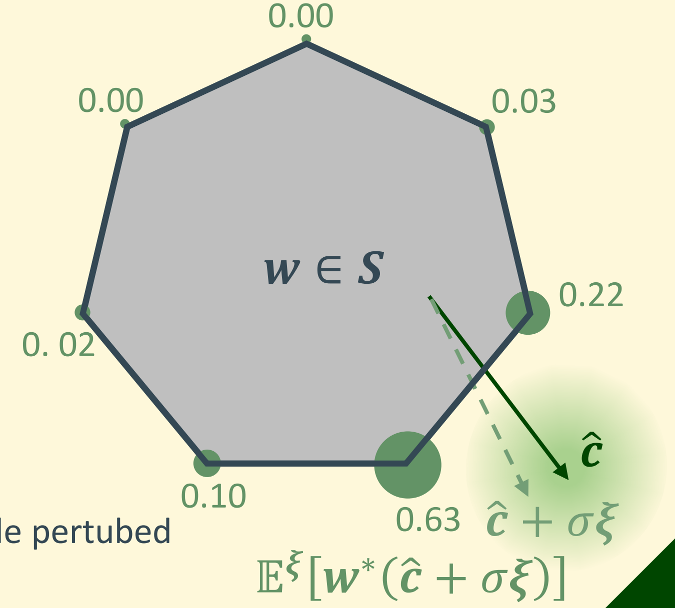
Use random perturbation to deal with the cost vector $\hat{\boldsymbol{c}}$.

The expected value $\mathbb{E}^{\xi}[\boldsymbol{w}^*(\hat{\boldsymbol{c}} + \sigma\boldsymbol{\xi})]$ repalces $\boldsymbol{w}^*(\hat{\boldsymbol{c}})$, which is the weighted average (convex combination) of the extreme points of feasible region.

How to calculate the expectation?

$$\mathbb{E}^{\xi}[\boldsymbol{w}^*(\hat{\boldsymbol{c}} + \sigma\boldsymbol{\xi})] \approx \frac{1}{\mathrm{K}}\sum_{\kappa}^{\mathrm{K}} \boldsymbol{w}^*(\hat{\boldsymbol{c}} + \sigma\boldsymbol{\xi}_{\boldsymbol{\kappa}})$$



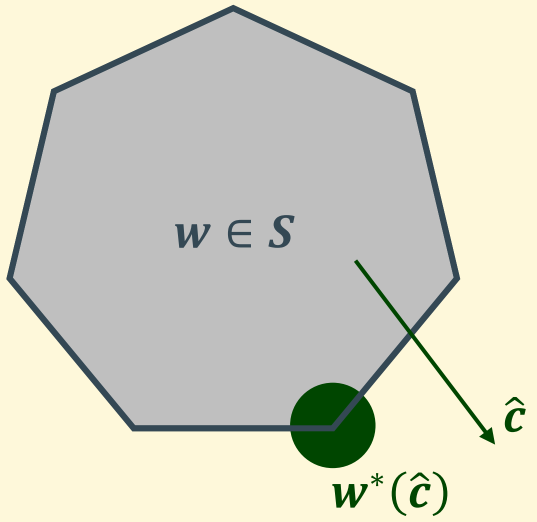


- Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J. P., & Bach, F. (2020). Learning with differentiable pertubed optimizers. Advances in neural information processing systems, 33, 9508-9519.
- Dalle, G., Baty, L., Bouvier, L., & Parmentier, A. (2022). Learning with combinatorial optimization layers: a probabilistic approach. arXiv preprint arXiv:2207.13513.c

Perturbed Method

Use random perturbation to deal with the cost vector $\hat{\boldsymbol{c}}$.

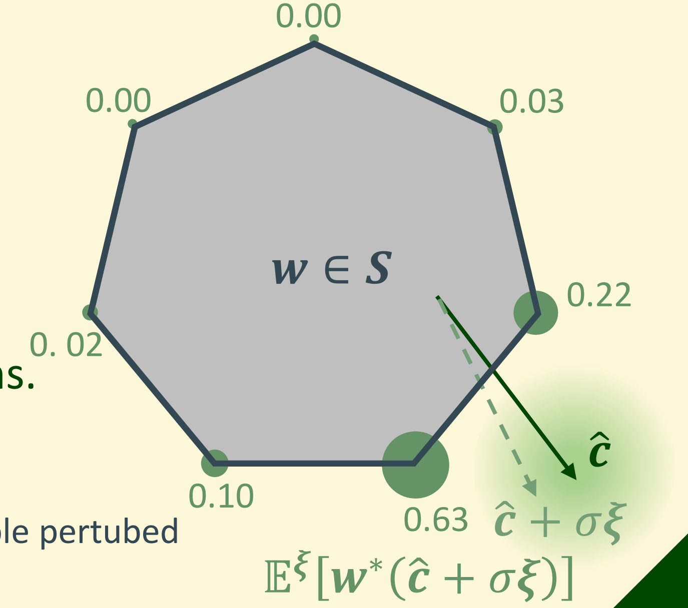
The expected value $\mathbb{E}^{\xi}[\boldsymbol{w}^*(\hat{\boldsymbol{c}} + \sigma\boldsymbol{\xi})]$ repalces $\boldsymbol{w}^*(\hat{\boldsymbol{c}})$, which is the weighted average (convex combination) of the extreme points of feasible region.



How to calculate the expectation?

$$\mathbb{E}^{\xi}[\boldsymbol{w}^*(\hat{\boldsymbol{c}} + \sigma\boldsymbol{\xi})] \approx \frac{1}{\mathrm{K}}\sum_{\kappa}^{\mathrm{K}} \underbrace{\boldsymbol{w}^*(\hat{\boldsymbol{c}} + \sigma\boldsymbol{\xi}_{\boldsymbol{\kappa}})}$$

Computational overhead: We need to solve K optimization problems.

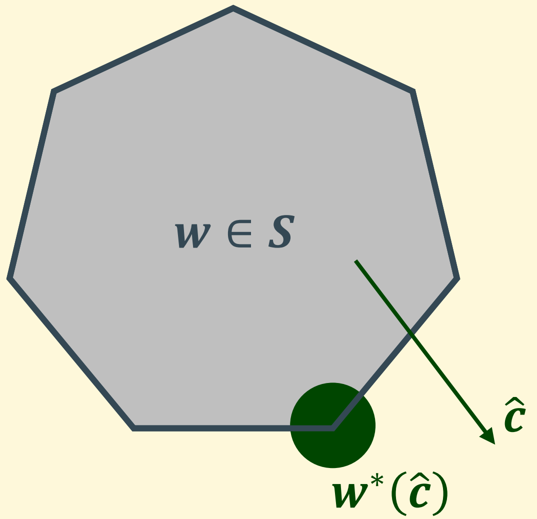


- Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J. P., & Bach, F. (2020). Learning with differentiable pertubed optimizers. Advances in neural information processing systems, 33, 9508-9519.
- Dalle, G., Baty, L., Bouvier, L., & Parmentier, A. (2022). Learning with combinatorial optimization layers: a probabilistic approach. arXiv preprint arXiv:2207.13513.c

Perturbed Method

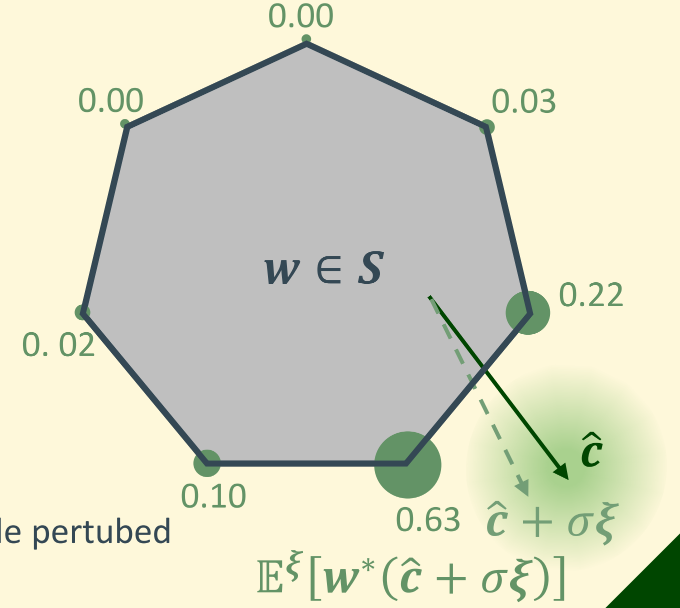
Use random perturbation to deal with the cost vector $\hat{\boldsymbol{c}}$.

The expected value $\mathbb{E}^{\xi}[\boldsymbol{w}^*(\hat{\boldsymbol{c}} + \sigma\boldsymbol{\xi})]$ repalces $\boldsymbol{w}^*(\hat{\boldsymbol{c}})$, which is the weighted average (convex combination) of the extreme points of feasible region.



How to calculate the expectation?

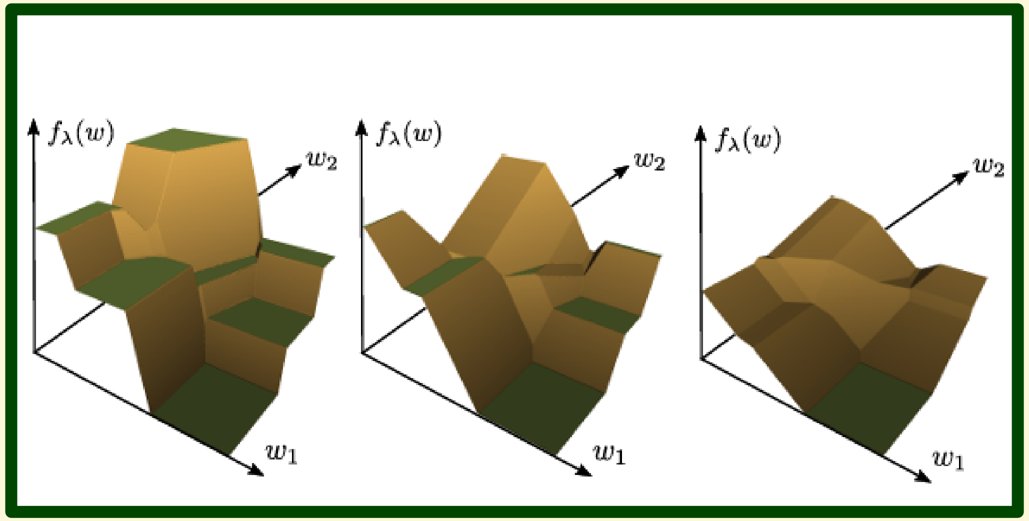
$$\mathbb{E}^{\xi}[\boldsymbol{w}^*(\hat{\boldsymbol{c}} + \sigma\boldsymbol{\xi})] \approx \frac{1}{\mathrm{K}}\sum_{\kappa}^{\mathrm{K}} \boldsymbol{w}^*(\hat{\boldsymbol{c}} + \sigma\boldsymbol{\xi}_{\boldsymbol{\kappa}})$$



A non-negativity requirement for $\hat{\boldsymbol{c}}$: multiplication perturbation

- Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J. P., & Bach, F. (2020). Learning with differentiable pertubed optimizers. Advances in neural information processing systems, 33, 9508-9519.
- Dalle, G., Baty, L., Bouvier, L., & Parmentier, A. (2022). Learning with combinatorial optimization layers: a probabilistic approach. arXiv preprint arXiv:2207.13513.c

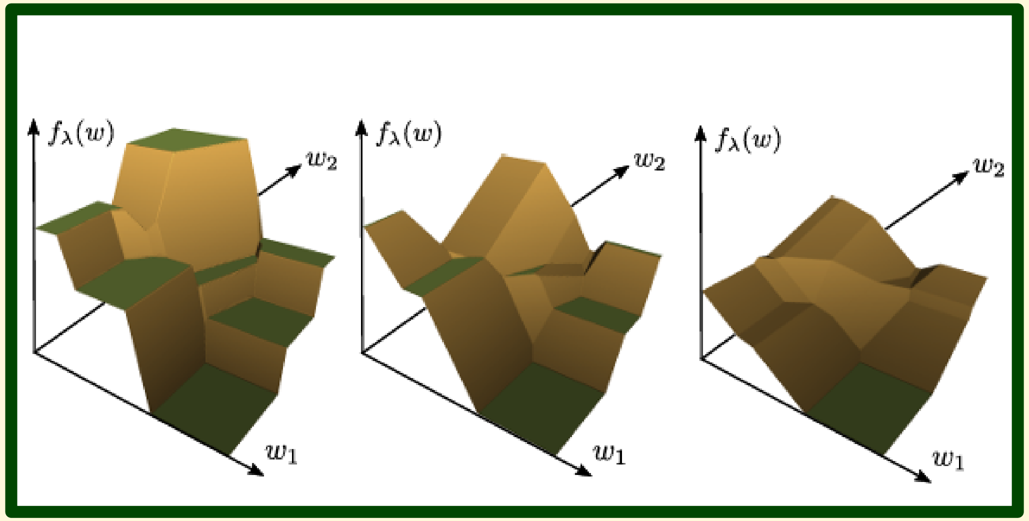
Black-box Method



- **Differentiable Black-box:** Perform continuous interpolation on the piecewise constant loss function to transform it into a piecewise linear function.

- Pogančić, M. V., Paulus, A., Musil, V., Martius, G., & Rolinek, M. (2019, September). Differentiation of blackbox combinatorial solvers. In International Conference on Learning Representations.
- Sahoo, S. S., Paulus, A., Vlastelica, M., Musil, V., Kuleshov, V., & Martius, G. (2022). Backpropagation through combinatorial algorithms: Identity with projection works. arXiv preprint arXiv:2205.15213.

Black-box Method



- **Differentiable Black-box:** Perform continuous interpolation on the piecewise constant loss function to transform it into a piecewise linear function.
- **Straight-Through Estimator:** Replace the solver gradient $\frac{\partial \boldsymbol{w}^*(\hat{\boldsymbol{c}})}{\partial \hat{\boldsymbol{c}}}$ with the negative identity matrix $-\boldsymbol{I}$.

- Pogančić, M. V., Paulus, A., Musil, V., Martius, G., & Rolinek, M. (2019, September). Differentiation of blackbox combinatorial solvers. In International Conference on Learning Representations.
- Sahoo, S. S., Paulus, A., Vlastelica, M., Musil, V., Kuleshov, V., & Martius, G. (2022). Backpropagation through combinatorial algorithms: Identity with projection works. arXiv preprint arXiv:2205.15213.

Contrastive & Ranking Method

During the training process, we can naturally collect a large number of feasible solutions, forming a solution set Γ.

- Mulamba, M., Mandi, J., Diligenti, M., Lombardi, M., Bucarey, V., & Guns, T. (2021). Contrastive losses and solution caching for predict-and-optimize. Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence.
- Mandi, J., Bucarey, V., Mulamba, M., & Guns, T. (2022). Decision-focused learning: through the lens of learning to rank. Proceedings of the 39th International Conference on Machine Learning.

Contrastive & Ranking Method

During the training process, we can naturally collect a large number of feasible solutions, forming a solution set Γ.

- **Contrastive Method:**

Take the subset of suboptimal solutions $\Gamma \backslash \boldsymbol{w}^*(\boldsymbol{c})$ as **negative samples**, to maximize the difference between the **optimal solution** and the **suboptimal solutions**.

$$\mathcal{L}_{NCE}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = \frac{1}{|\Gamma| - 1} \sum_{\Gamma \backslash \boldsymbol{w}^*(\boldsymbol{c})}^{\boldsymbol{w}^\gamma} (\hat{\boldsymbol{c}}^T \boldsymbol{w}^*(\boldsymbol{c}) - \hat{\boldsymbol{c}}^T \boldsymbol{w}^\gamma)$$

- Mulamba, M., Mandi, J., Diligenti, M., Lombardi, M., Bucarey, V., & Guns, T. (2021). Contrastive losses and solution caching for predict-and-optimize. Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence.
- Mandi, J., Bucarey, V., Mulamba, M., & Guns, T. (2022). Decision-focused learning: through the lens of learning to rank. Proceedings of the 39th International Conference on Machine Learning.

Contrastive & Ranking Method

During the training process, we can naturally collect a large number of feasible solutions, forming a solution set Γ.

- **Contrastive Method:**

Take the subset of suboptimal solutions $\Gamma \backslash \boldsymbol{w}^*(\boldsymbol{c})$ as **negative samples**, to maximize the difference between the **optimal solution** and the **suboptimal solutions**.

$$\mathcal{L}_{NCE}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = \frac{1}{|\Gamma| - 1} \sum_{\Gamma \backslash \boldsymbol{w}^*(\boldsymbol{c})}^{\boldsymbol{w}^\gamma} (\hat{\boldsymbol{c}}^T \boldsymbol{w}^*(\boldsymbol{c}) - \hat{\boldsymbol{c}}^T \boldsymbol{w}^\gamma)$$

- **Ranking Method:**

Transform the predict-then-optimize task as Learning to Rank, with the **objective value** (such as $\hat{\boldsymbol{c}}^T \boldsymbol{w}$) as score, in order to correctly rank the subset of feasible solutions Γ.

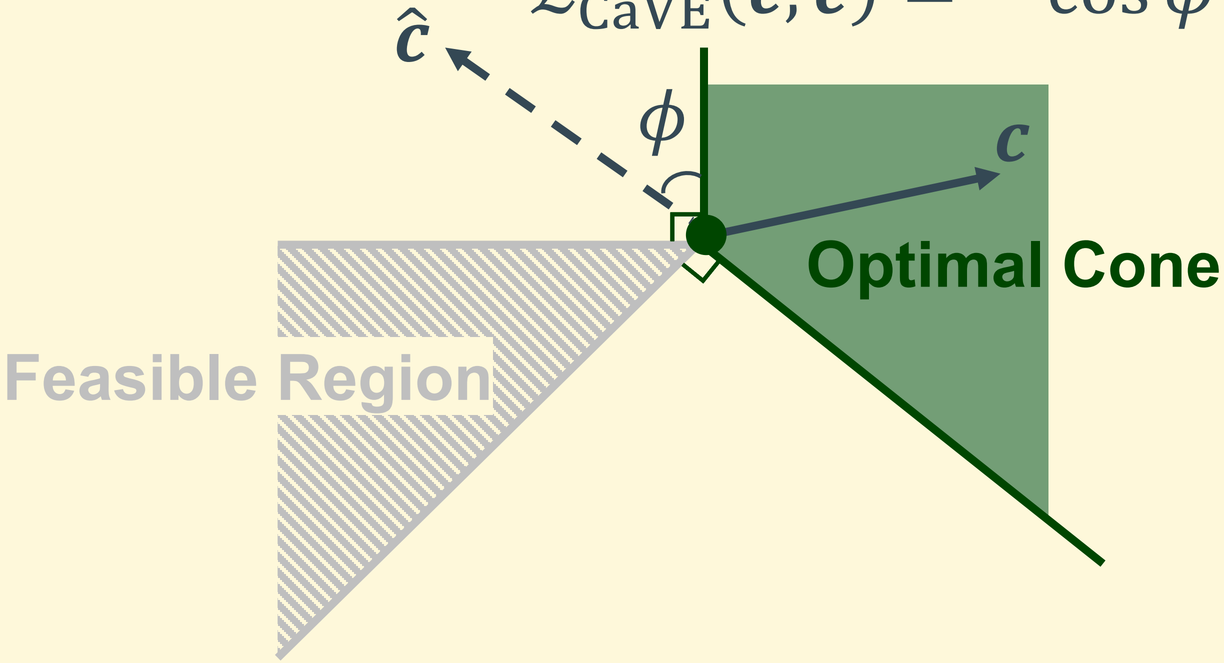
- Mulamba, M., Mandi, J., Diligenti, M., Lombardi, M., Bucarey, V., & Guns, T. (2021). Contrastive losses and solution caching for predict-and-optimize. Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence.
- Mandi, J., Bucarey, V., Mulamba, M., & Guns, T. (2022). Decision-focused learning: through the lens of learning to rank. Proceedings of the 39th International Conference on Machine Learning.

Projection Method

- The loss of CaVE employs cosine similarity to minimize the angle ϕ between the predicted costs $\hat{\boldsymbol{c}}$ and a normal cone of optimal solution.

- A quadratic programming is required to get a projection of prediction on the cone.

$$\mathcal{L}_{\mathrm{CaVE}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) = -\cos\phi$$



- Tang, B., & Khalil, E. B. (2024, May). CaVE: A Cone-Aligned Approach for Fast Predict-then-optimize with Binary Linear Programs. In International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (pp. 193-210). Cham: Springer Nature Switzerland.

Software

GitHub

An open-sourced library to facilitate predict-then-optimize, bridging the gap between optimization and machine learning.

Software

Benchmarks

- Shortest Path
- Knapsack
- Traveling Salesperson

Opt Modeling

- PYOMO
- GUROBI OPTIMIZATION
- COPT Cardinal Optimizer
- Any custom solver

ML Modeling

- PyTorch
- scikit learn

Algorithms

- SPO+
- DBB
- NID
- PFYL
- NCE
- LTR
- ...

Autograd Function

Algorithms

- SPO+
- DBB
- NID
- DPO
- PFYL
- NCE
- LTR

`pyepo.func.perturbedFenchelYoung` allows us to set a Fenchel-Young loss for training, which requires parameters:

- `optmodel` : a PyEPO optimization model
- `n_samples` : number of Monte-Carlo samples
- `sigma` : the amplitude of the perturbation for costs
- `processes` : number of processors for multi-thread, 1 for single-core, 0 for all of the cores
- `seed` : random state seed for perturbations

```
import pyepo

# init SPO+ loss
spop = pyepo.func.SPOPlus(optmodel, processes=2)
# init PFY loss
pfy = pyepo.func.perturbedFenchelYoung(optmodel, n_samples=3, sigma=1.0, processes=2)
# init NCE loss
nce = pyepo.func.NCE(optmodel, processes=2, solve_ratio=0.05, dataset=dataset_train)
```

Modeling

```python
import gurobipy as gp
from gurobipy import GRB
from pyepo.model.grb import optGrbModel

class myOptModel(optGrbModel):
    def _getModel(self):
        # ceate a model
        m = gp.Model()
        # varibles
        x = m.addVars(5, name="x", vtype=GRB.BINARY)
        # sense
        m.modelSense = GRB.MAXIMIZE
        # constraints
        m.addConstr(3*x[0]+4*x[1]+3*x[2]+6*x[3]+4*x[4]<=12)
        m.addConstr(4*x[0]+5*x[1]+2*x[2]+3*x[3]+5*x[4]<=10)
        m.addConstr(5*x[0]+4*x[1]+6*x[2]+2*x[3]+3*x[4]<=15)
        return m, x

optmodel = myOptModel()
```

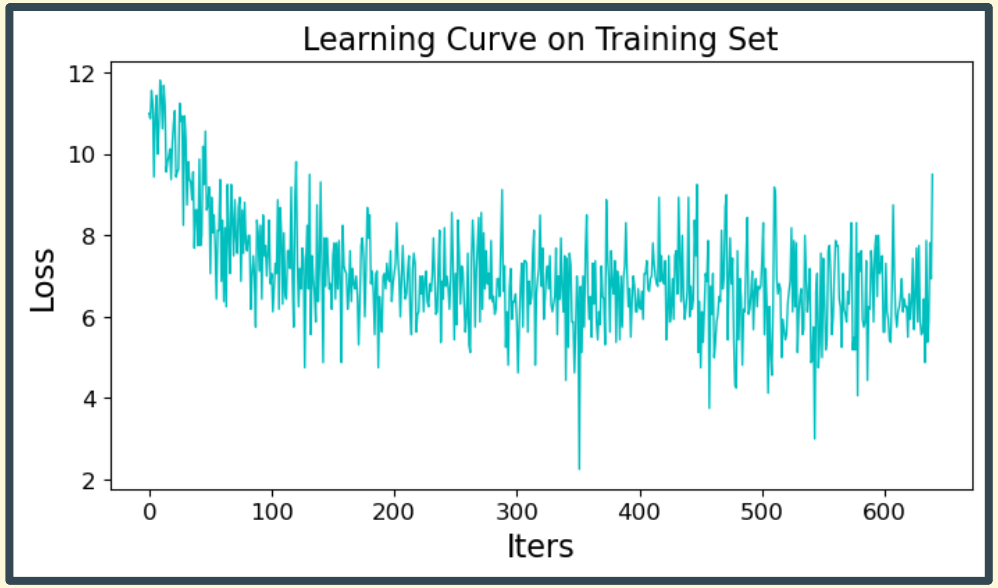
$$\max_{w} \sum_{i=0}^{4} c_i w_i$$

$$\begin{aligned} s.t.\quad & 3w_0 + 4w_1 + 3w_2 + 6w_3 + 4w_4 \leq 12 \\ & 4w_0 + 5w_1 + 2w_2 + 3w_3 + 5w_4 \leq 10 \\ & 5w_0 + 4w_1 + 6w_2 + 2w_3 + 3w_4 \leq 10 \\ & w_0, w_1, w_2, w_3, w_4 \in \{0,1\} \end{aligned}$$

Optimization Model

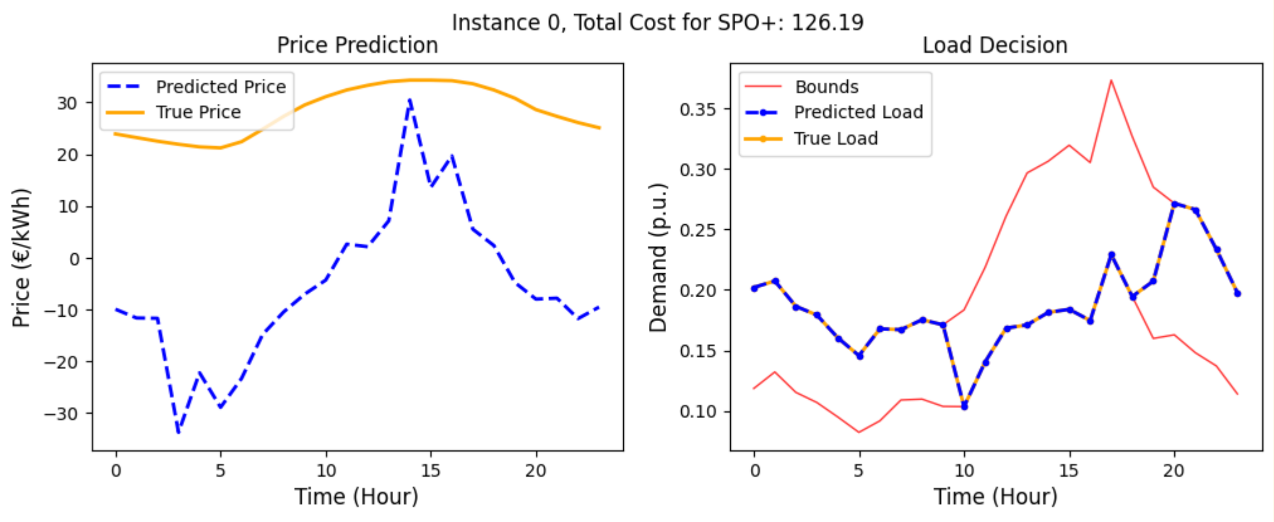
End-to-End Training

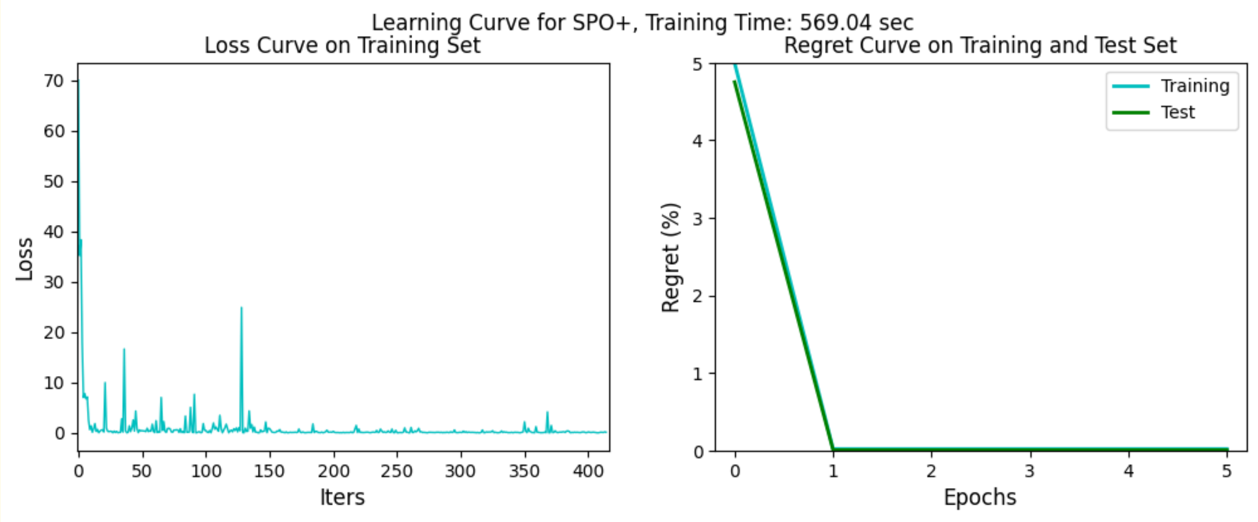
```
# set adam optimizer
optimizer = torch.optim.Adam(reg.parameters(), lr=5e-3)

# train mode
reg.train()
for epoch in range(5):
  # load data
  for i, data in enumerate(loader_train):
      x, c, w, z = data # feat, cost, sol, obj
      # cuda
      if torch.cuda.is_available():
          x, c, w, z = x.cuda(), c.cuda(), w.cuda(), z.cuda()
      # forward pass
      cp = reg(x)
      loss = pfy(cp, w)
      # backward pass
      optimizer.zero_grad()
      loss.backward()
      optimizer.step()
```



2D knapsack Visualization

Real-world Energy Scheduling






```
Warning: Cannot load hourly/2022/08329.csv.gz from https://bulk.meteostat.net/v2/
Warning: Cannot load hourly/2023/08329.csv.gz from https://bulk.meteostat.net/v2/
Warning: Cannot load hourly/2022/69669.csv.gz from https://bulk.meteostat.net/v2/
Warning: Cannot load hourly/2023/69669.csv.gz from https://bulk.meteostat.net/v2/
Warning: Cannot load hourly/2022/LEZA0.csv.gz from https://bulk.meteostat.net/v2/
Warning: Cannot load hourly/2023/LEZA0.csv.gz from https://bulk.meteostat.net/v2/
```

time	temp	dwpt	rhum	prcp	snow	wdir	wspd	wpgt	pres	tsun	coco
2022-01-01 00:00:00	10.1	5.0	75.2	0.0	NaN	160.3	7.9	20.9	1026.9	0.0	2.2
2022-01-01 01:00:00	9.9	4.6	74.7	0.0	NaN	158.1	8.0	22.0	1026.7	0.0	2.2
2022-01-01 02:00:00	9.6	4.5	75.2	0.0	NaN	168.2	7.9	21.6	1026.7	0.0	2.1
2022-01-01 03:00:00	9.4	4.0	74.0	0.0	NaN	156.0	7.9	21.8	1026.7	0.0	2.4
2022-01-01 04:00:00	9.1	3.7	73.8	0.0	NaN	154.6	7.7	20.9	1026.6	0.0	2.4

Real-World Energy Scheduling

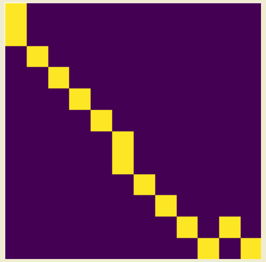
Predicting Shortest Paths from Images

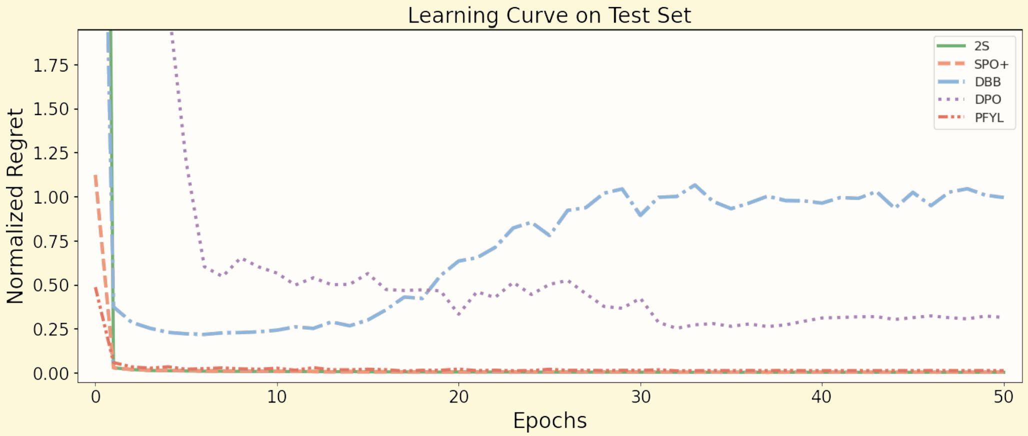
Warcraft Dataset [Pogančić et al. (2019)]

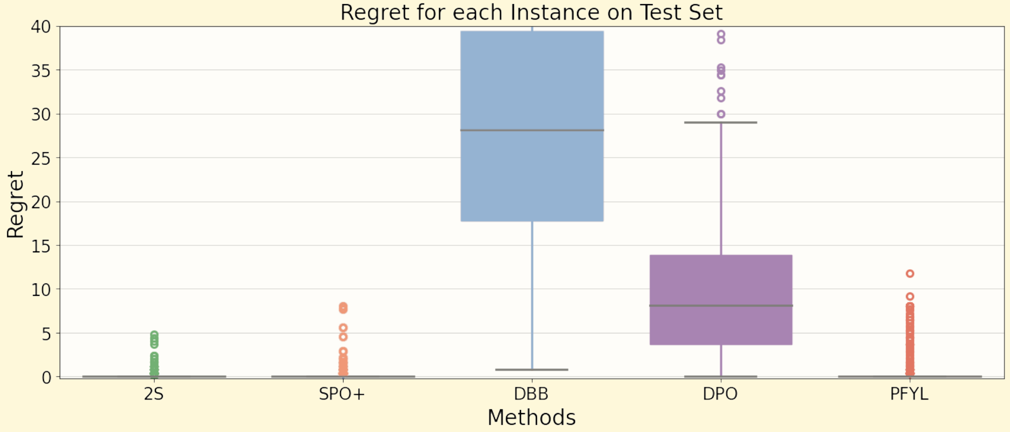
- Terrain Image

- Vertex Weights

- **Shortest Path**







[Warcraft Shortest Path]

Thank You

GitHub

Mechanical & Industrial Engineering
UNIVERSITY OF TORONTO