Scalability Challenges and Solutions to Differentiable Optimization

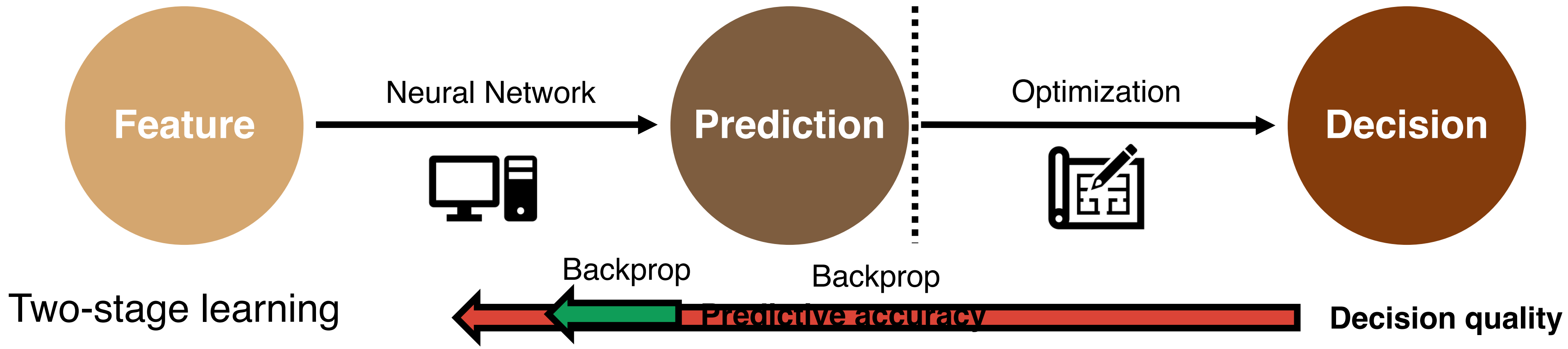
2022/07/25 @ IJCAI 2022

Kai Wang

Harvard University

Decision-focused Learning (Recap)

Maximize **decision quality** directly by integrating optimization as a differentiable layer



Public health

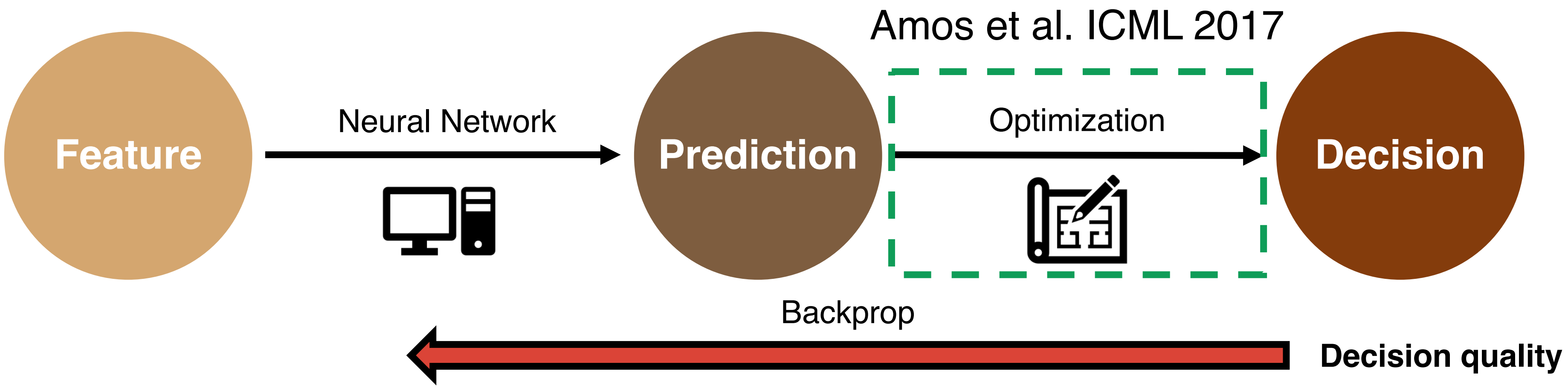
Route planning

Movie recommendation

Portfolio optimization

Decision-focused Learning (Recap)

Maximize **decision quality** directly by integrating optimization as a differentiable layer



Research questions

- **Scalability**: how to make decision-focused learning more efficient?
- **Extension**: how to extend to other optimization problems?

Outline

- **Decision-focused Learning (Recap)**
- **Scalability and Applications in Different Optimization Problems**
  - Convex/non-convex optimization
  - Sequential optimization
  - Multi-agent optimization
- **Summary of Differentiable Optimization**

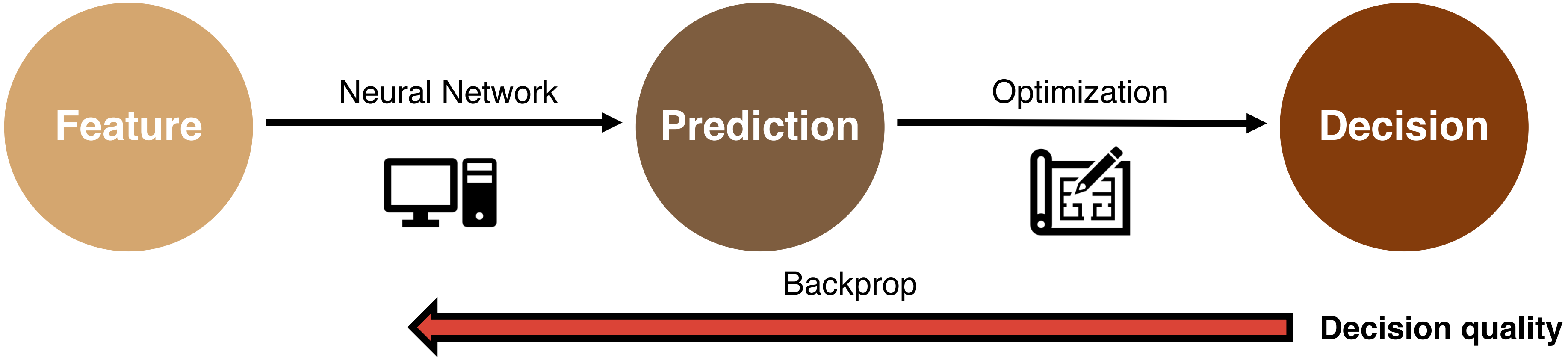
Outline

- Decision-focused Learning (Recap)
- **Scalability and Applications in Different Optimization Problems**
  - Convex/non-convex optimization
  - Sequential optimization
  - Multi-agent optimization
- Summary of Differentiable Optimization

Decision-focused Learning

AAMAS 2020

Convex/non-convex optimization



Decision-focused Learning

AAMAS 2020

Convex/non-convex optimization

Forward path: solving the planning problem

Expensive!!

Feature $\xrightarrow{\text{Neural Network}}$ Prediction θ $\xrightarrow{\text{Optimization}}$ Decision z^*

$$\max_z f(z,\theta) \quad s.t.\ Az = b,\ Gz \leq h$$

Backprop

Decision quality

Backward path: differentiating through the planning problem

Expensive!!

$$f(z,\theta) \approx f(z_0,\theta) + \nabla f \cdot \Delta z + \frac{1}{2}\Delta z^T \cdot \nabla^2 f \cdot \Delta z \coloneqq g(z,\theta)$$

Non-convex

$$\max_z f(z,\theta) \quad s.t.\ Az = b,\ Gz \leq h$$

local $\rightarrow$

Convex

$$\max_z g(z,\theta) \quad s.t.\ Az = b,\ Gz \leq h$$

CVXpy

$$x^*(\theta) = \underset{x}{\text{argmin}}\ f(x;\theta) \quad \text{subject to } g(x;\theta) \leq 0,\ h(x;\theta) = 0$$

PyTorch

TensorFlow

$$\frac{dz^*}{d\theta} = -\left(\boxed{\frac{\partial}{\partial z} KKT(z^*,\theta)}\right)^{-1} \frac{\partial}{\partial \theta} KKT(z^*,\theta)$$

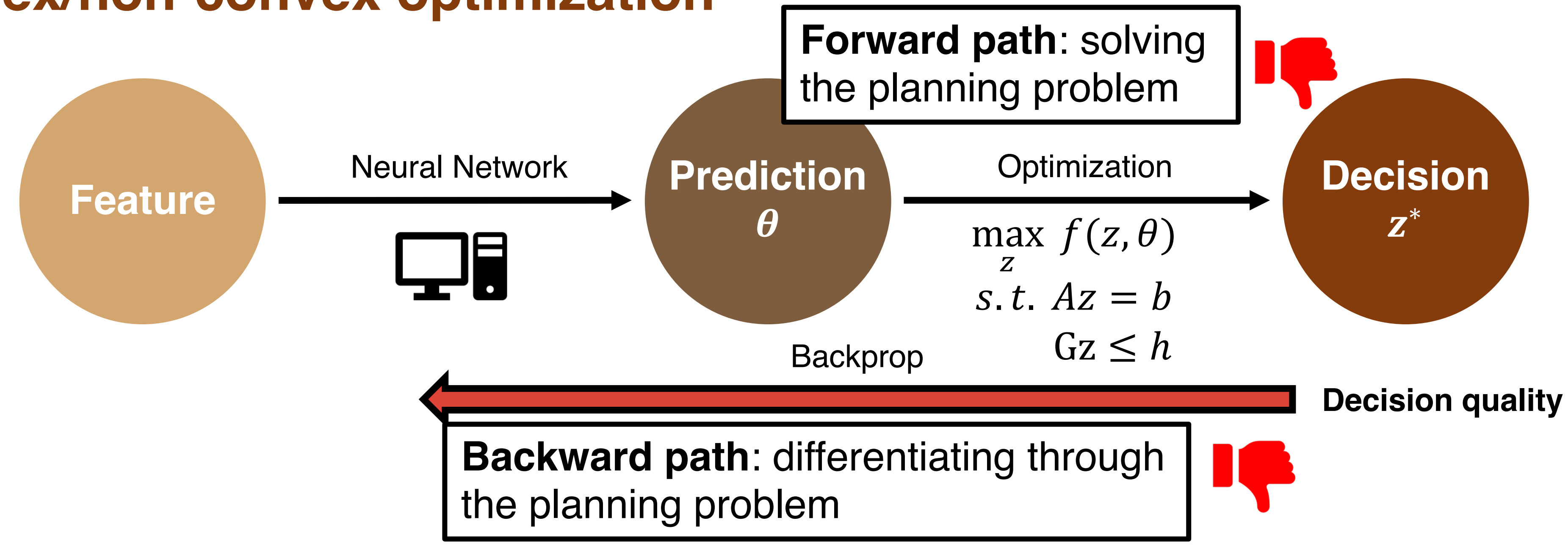
Second-order derivative with size $N \times N$

Matrix inversion cost $O(N^{\omega})$, $\omega \approx 2.373$

Surrogate Decision-focused Learning

NeurIPS 2020 spotlight

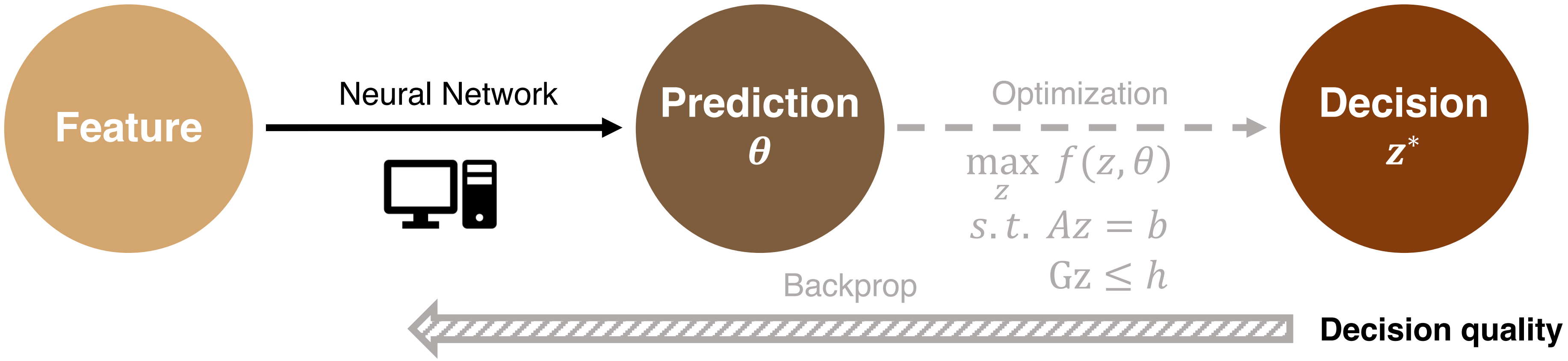
Convex/non-convex optimization



Surrogate Decision-focused Learning

NeurIPS 2020
spotlight

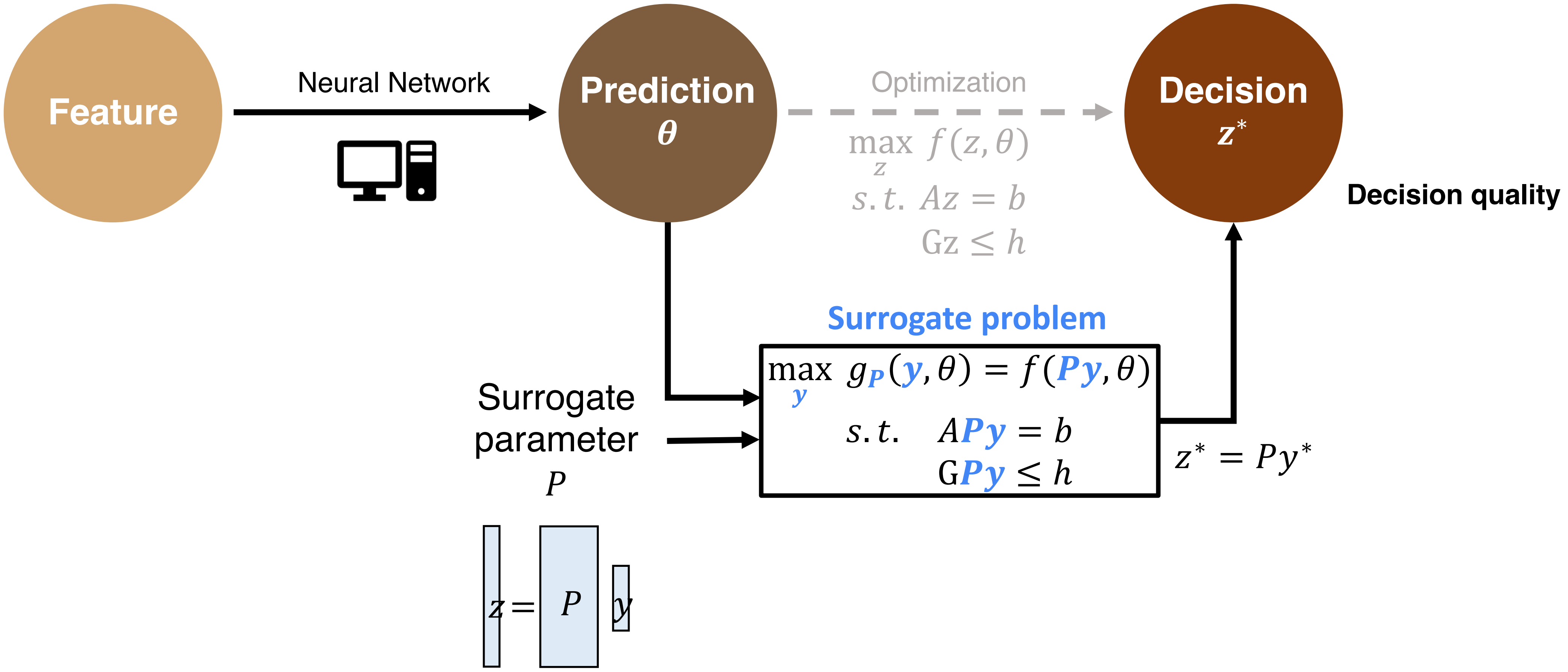
Convex/non-convex optimization



Surrogate Decision-focused Learning

NeurIPS 2020
spotlight

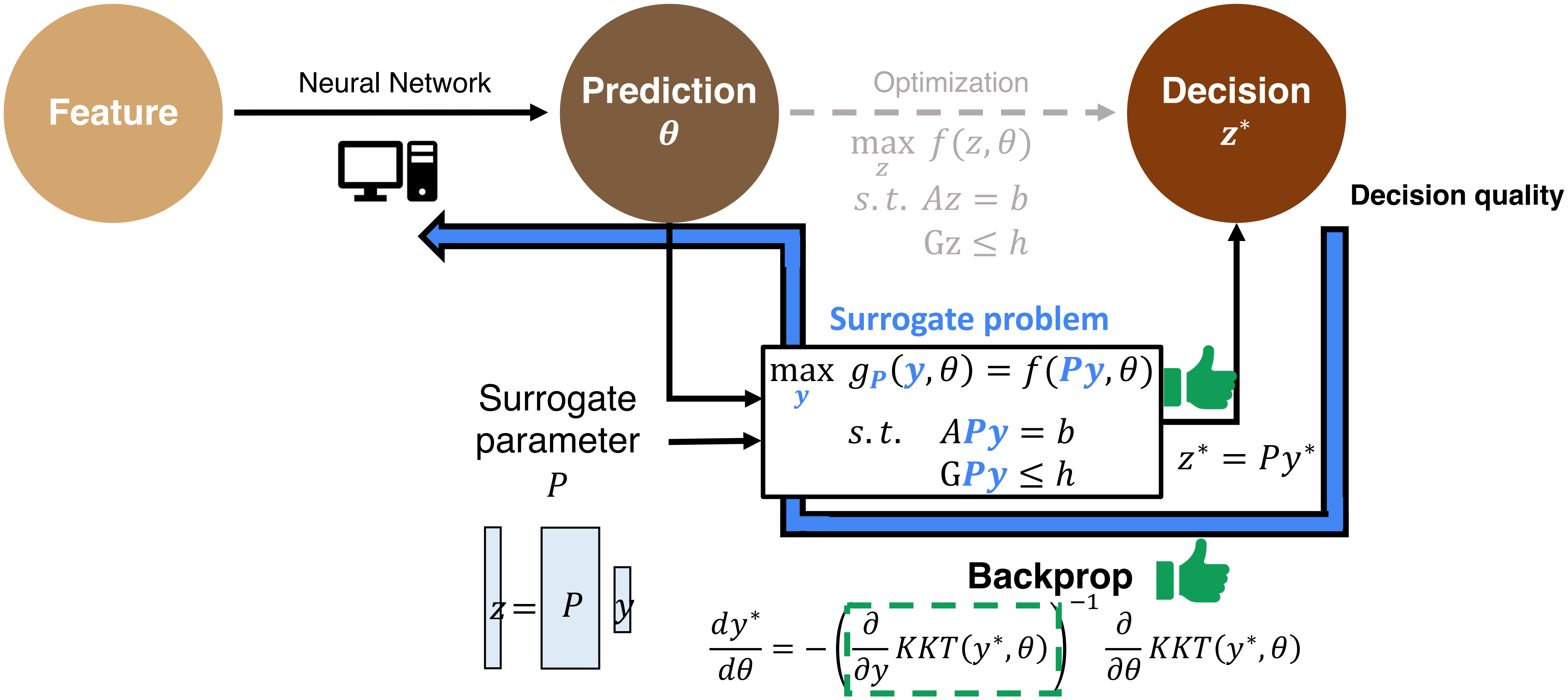
Convex/non-convex optimization



Surrogate Decision-focused Learning

NeurIPS 2020 spotlight

Convex/non-convex optimization



$$\frac{dy^*}{d\theta} = -\left(\frac{\partial}{\partial y} KKT(y^*,\theta)\right)^{-1} \frac{\partial}{\partial \theta} KKT(y^*,\theta)$$

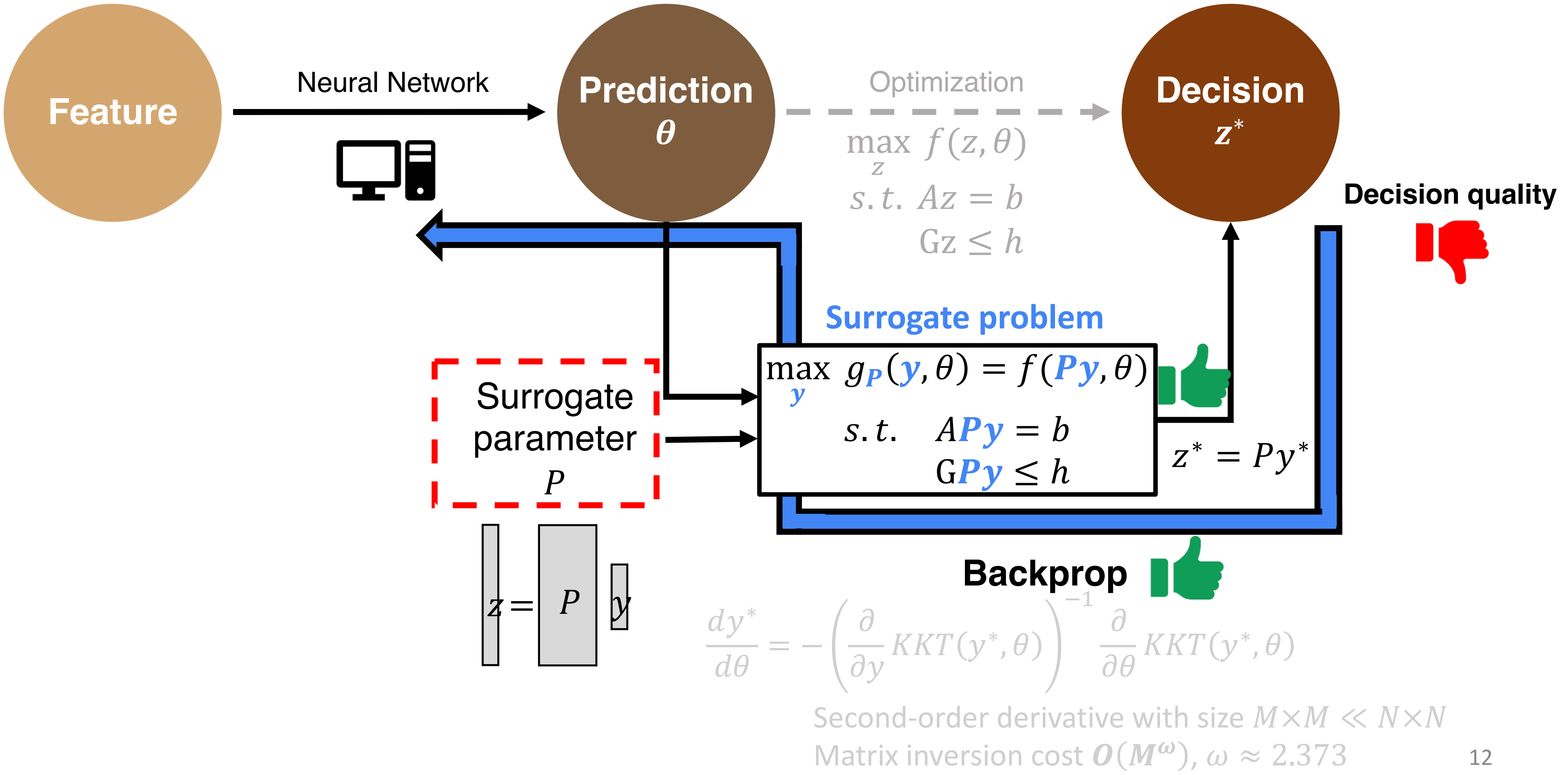
Second-order derivative with size $M\times M \ll N\times N$

Matrix inversion cost $O(M^{\omega})$, $\omega \approx 2.373$

Surrogate Decision-focused Learning

Convex/non-convex optimization

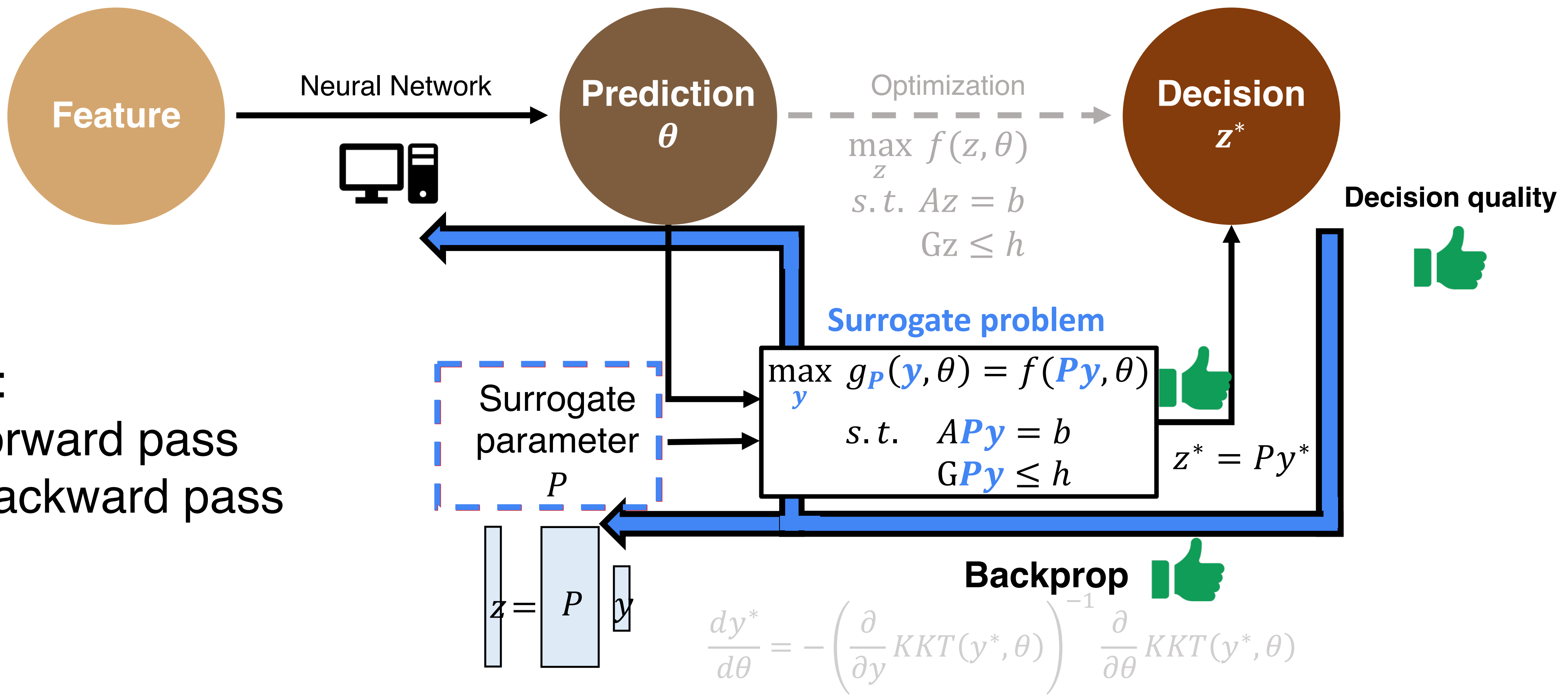
NeurIPS 2020 spotlight



Surrogate Decision-focused Learning

NeurIPS 2020
spotlight

Convex/non-convex optimization



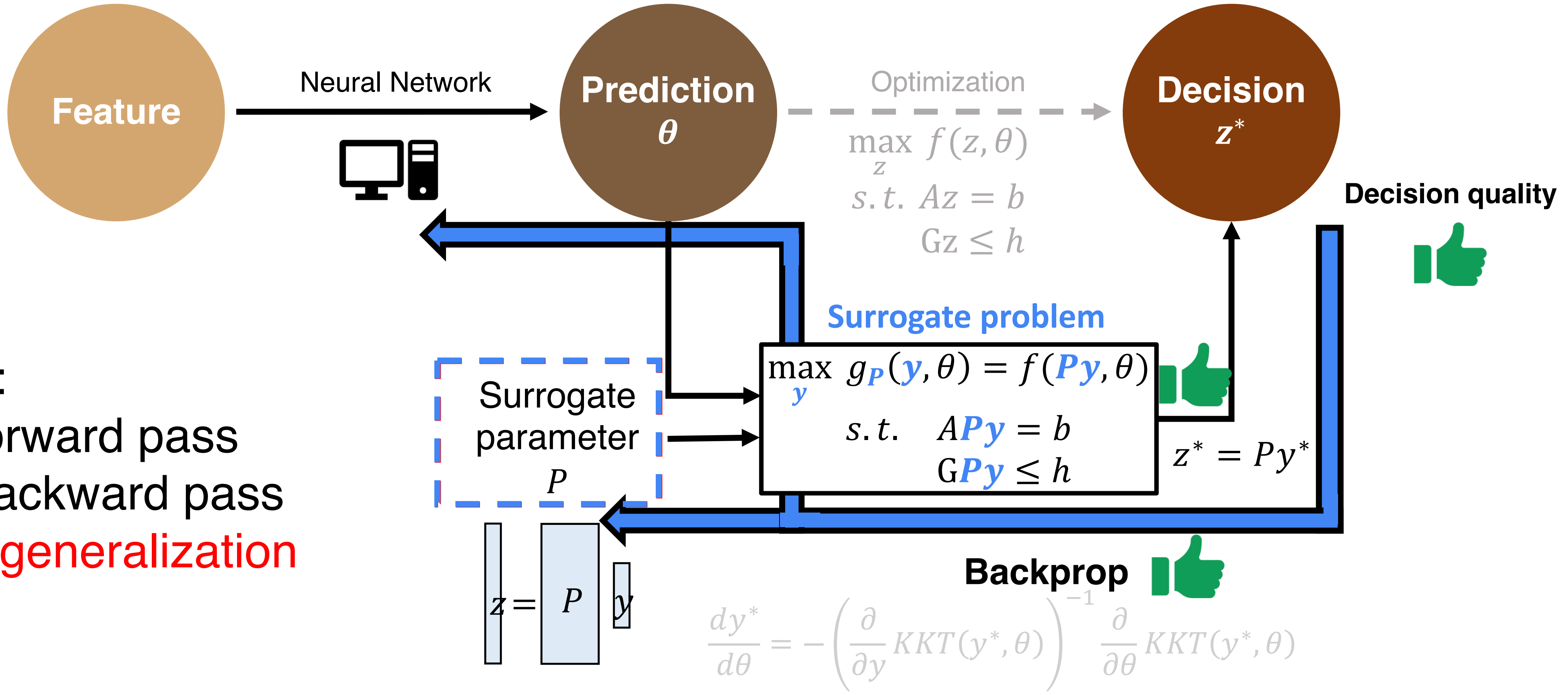
Benefits:
- Fast forward pass
- Fast backward pass

Second-order derivative with size $M \times M \ll N \times N$
Matrix inversion cost $O(M^{\omega})$, $\omega \approx 2.373$

Surrogate Decision-focused Learning

NeurIPS 2020
spotlight

Convex/non-convex optimization



Second-order derivative with size $M \times M \ll N \times N$

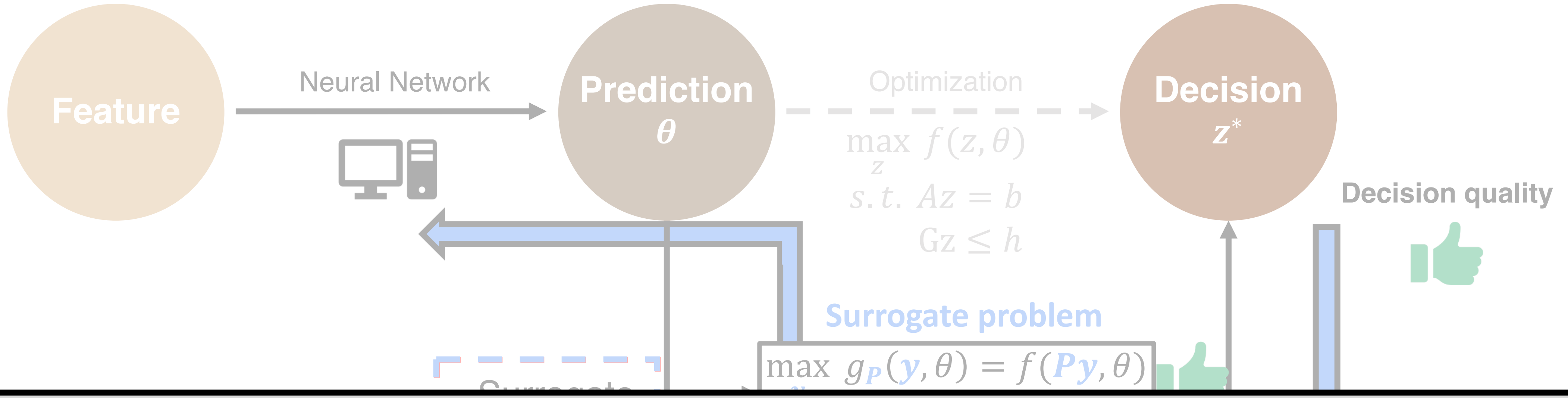
Matrix inversion cost $O(M^{\omega})$, $\omega \approx 2.373$

Benefits:
- Fast forward pass
- Fast backward pass
- Better generalization

Surrogate Decision-focused Learning

NeurIPS 2020
spotlight

Convex/non-convex optimization



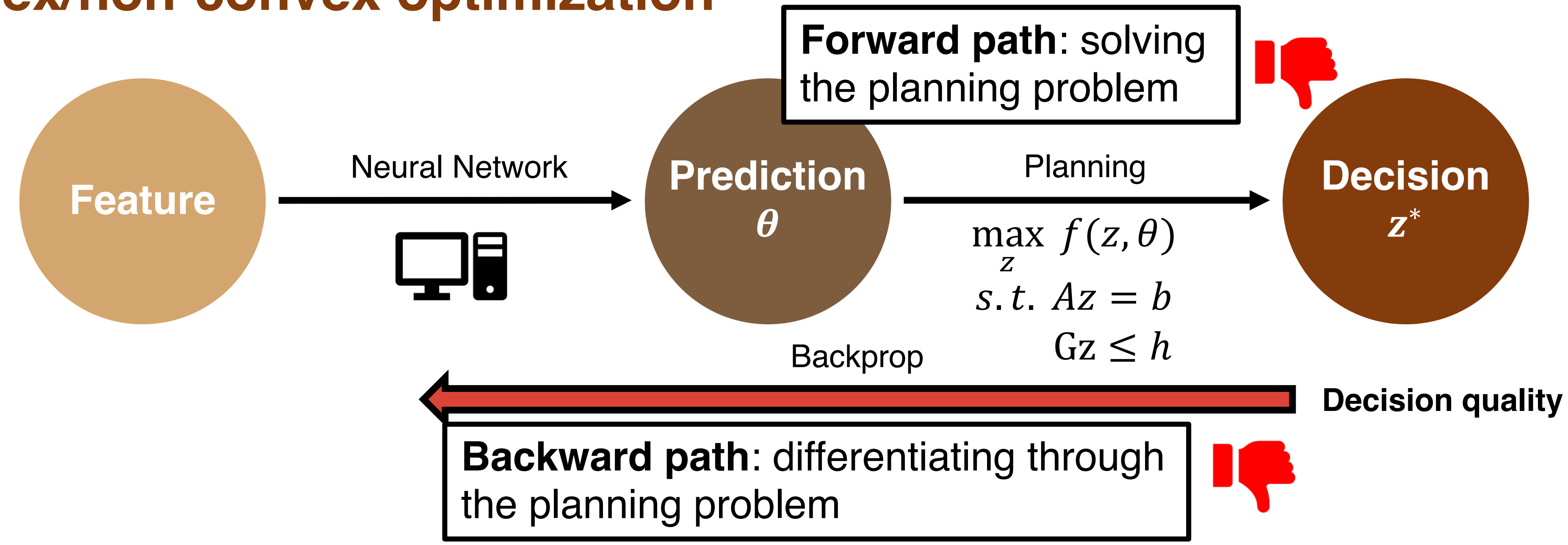
Takeaway:
- Improve scalability by **surrogate** and dimension reduction
- Jointly learn the **surrogate** and the model

Second-order derivative with size $M \times M \ll N \times N$

Matrix inversion cost $O(M^\omega)$, $\omega \approx 2.373$

Decision-focused Learning

Convex/non-convex optimization



$$\frac{dz^*}{d\theta} = -\left(\boxed{\frac{\partial}{\partial z} KKT(z^*,\theta)}\right)^{-1} \frac{\partial}{\partial \theta} KKT(z^*,\theta)$$

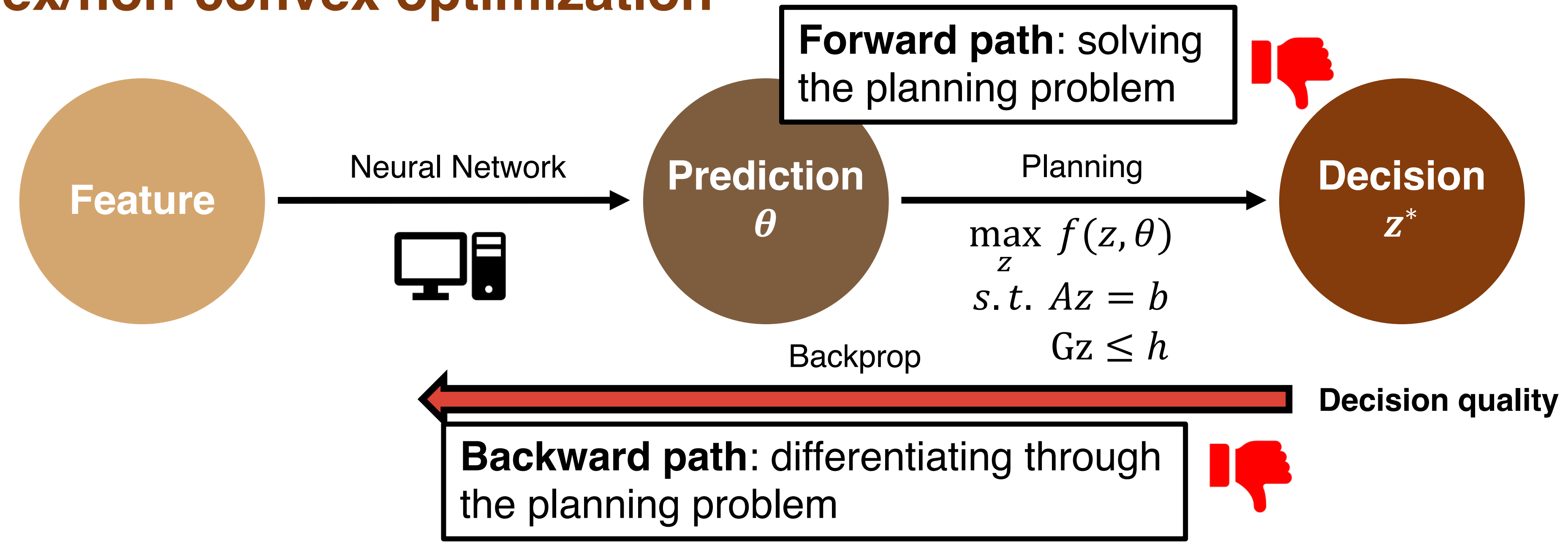
Second-order derivative involved with size N^2

Matrix inversion cost $O(N^\omega)$, $\omega \approx 2.373$

Block Decision-focused Learning

AAMAS 2020

Convex/non-convex optimization



$$\frac{dz^*}{d\theta} = -\left(\boxed{\frac{\partial}{\partial z} KKT(z^*, \theta)}\right)^{-1} \frac{\partial}{\partial \theta} KKT(z^*, \theta)$$

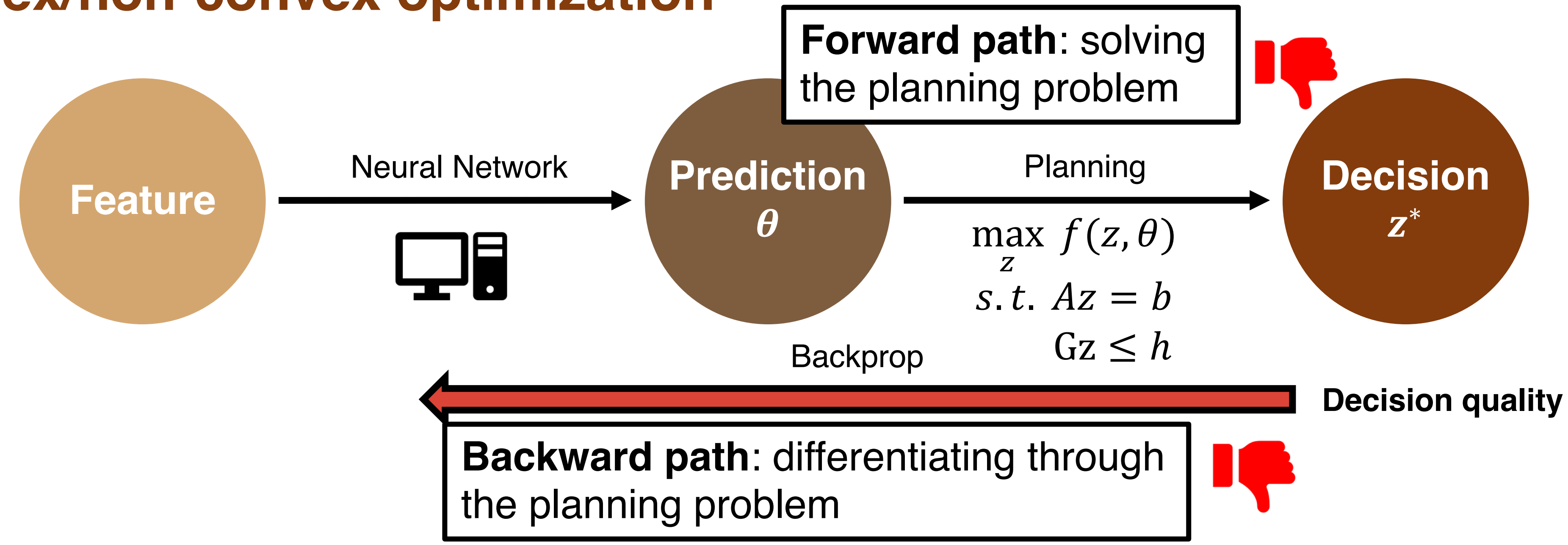
Second-order derivative involved with size N^2

Matrix inversion cost $O(N^\omega)$, $\omega \approx 2.373$

Block Decision-focused Learning

AAMAS 2020

Convex/non-convex optimization



Main idea: only backprop through a subset of decision variables

$$\frac{dz^*}{d\theta} = -\left(\boxed{\frac{\partial}{\partial z} KKT(z^*,\theta)}\right)^{-1} \frac{\partial}{\partial \theta} KKT(z^*,\theta)$$

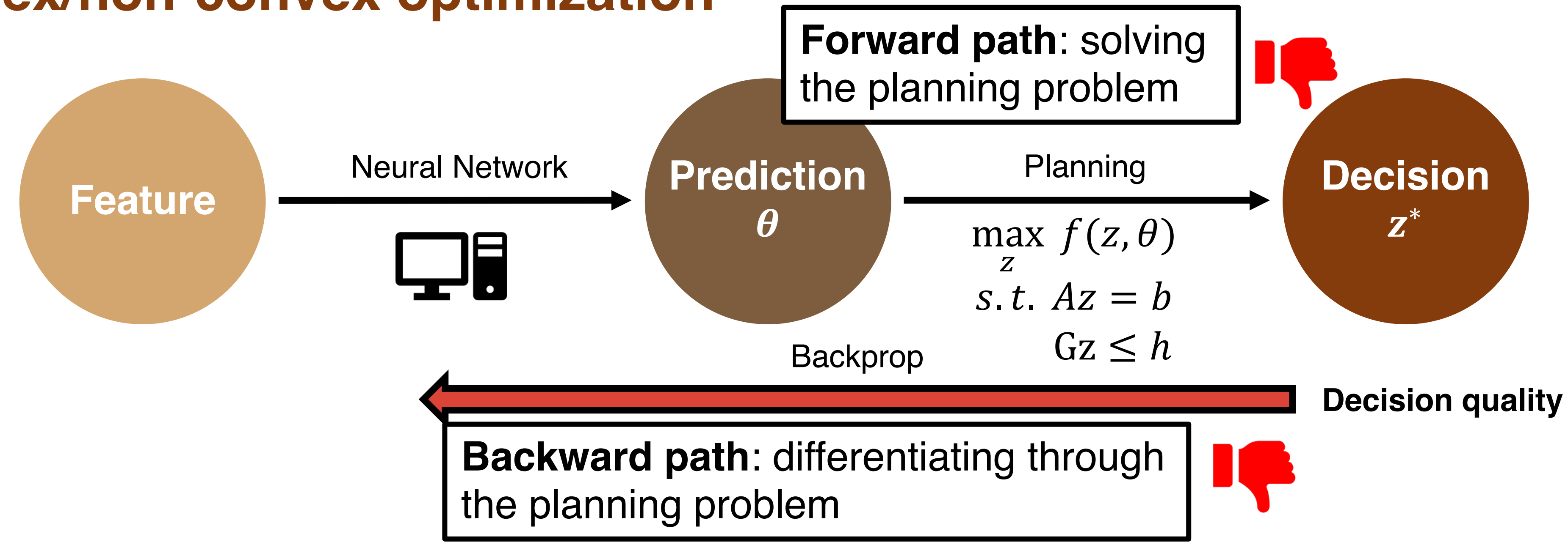
Second-order derivative involved with size N^2

Matrix inversion cost $O(N^{\omega})$, $\omega \approx 2.373$

Block Decision-focused Learning

AAMAS 2020

Convex/non-convex optimization

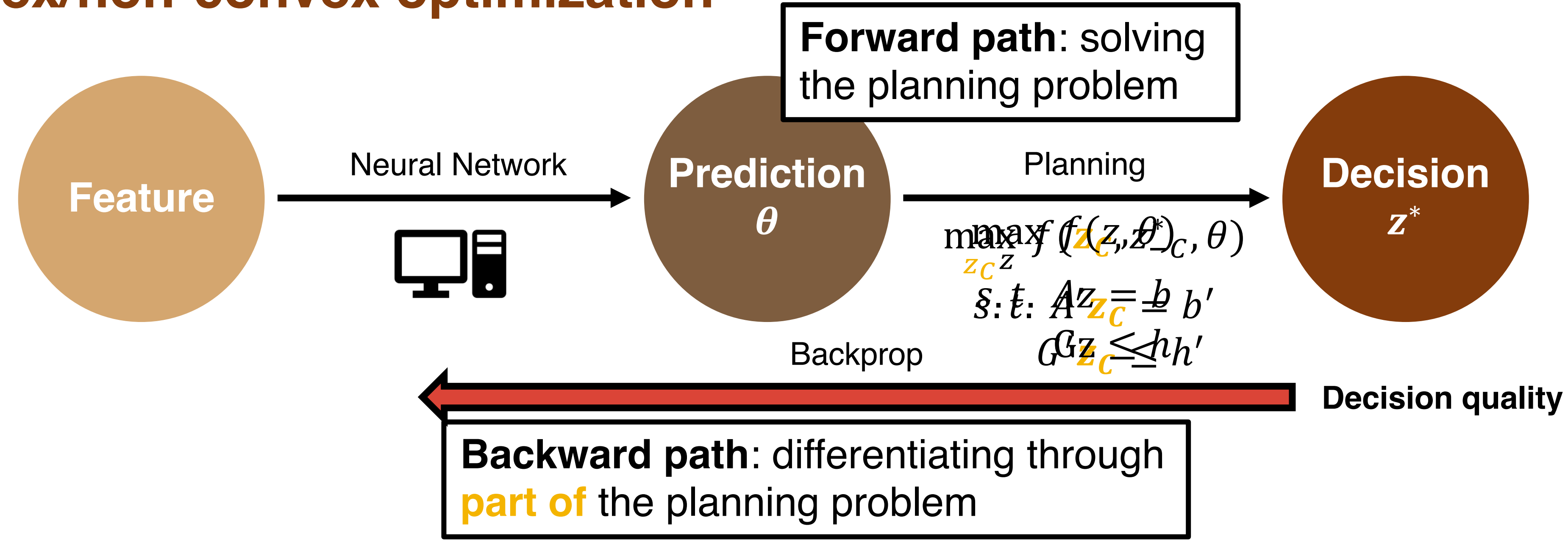


Main idea: only backprop through a **subset** of decision variables

Block Decision-focused Learning

AAMAS 2020

Convex/non-convex optimization



Algorithm:

$$\frac{dz^*_C}{d\theta} = -\left(\frac{\partial}{\partial z_C} KKT_C(z^*, \theta)\right)^{-1} \frac{\partial}{\partial \theta} KKT_C(z^*, \theta)$$

1. Solve z (forward)
2. Randomly sample a subset C
3. Compute $\frac{dz^*_C}{d\theta}$ to backprop (backward)

Block Decision-focused Learning

AAMAS 2020

Convex/non-convex optimization

Forward path: solving the planning problem

Feature → Neural Network → **Prediction** θ → Planning → **Decision** z^*

$$\max_{z_C} f(z_C, z^*_{-C}, \theta)$$
$$s.t.\ A' z_C = b'$$
$$G' z_C \le h'$$

Backprop

Decision quality

Backward path: differentiating through part of the planning problem

Algorithm:

1. Solve z (forward)
2. Randomly sample a subset C
3. Compute $\frac{dz_C^*}{d\theta}$ to backprop (backward)

$$\frac{dz_C^*}{d\theta} = -\boxed{\left(\frac{\partial}{\partial z_C} KKT_C(z^*, \theta)\right)}^{-1} \frac{\partial}{\partial \theta} KKT_C(z^*, \theta)$$

Second-order derivative involved with size $|C|^2$

Matrix inversion cost $O(|C|^\omega)$, $\omega \approx 2.373$

Block Decision-focused Learning

AAMAS 2020

Convex/non-convex optimization

Forward path: solving the planning problem

Feature → (Neural Network) → Prediction θ → (Planning) → Decision z^*

$$\max_{z_C} f(z_C, z^*_{-C}, \theta)$$
$$s.t.\ A' z_C = b'$$
$$G' z_C \leq h'$$

Backprop

Decision quality

Backward path: differentiating through part of the planning problem

$$\boxed{\frac{dz^*_C}{d\theta}} = -\boxed{\left(\frac{\partial}{\partial z_C} KKT_C(z^*, \theta)\right)^{-1}} \frac{\partial}{\partial \theta} KKT_C(z^*, \theta)$$

Second-order derivative involved with size $|C|^2$

Matrix inversion cost $O(|C|^\omega)$, $\omega \approx 2.373$

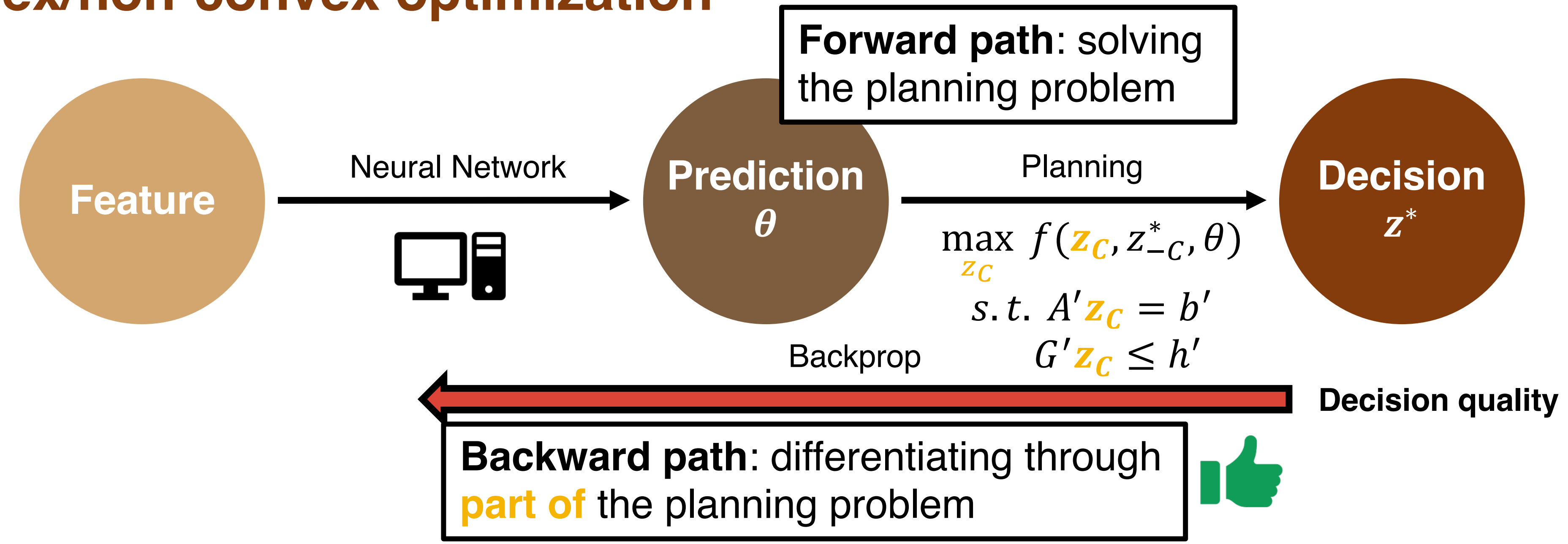
Observations:

- More than quadratic speedup
- Linearly less information
- Approximate stochastic gradient

Block Decision-focused Learning

AAMAS 2020

Convex/non-convex optimization



Takeaway:

- Improve scalability by **block sampling**
- Approximate stochastic gradient descent

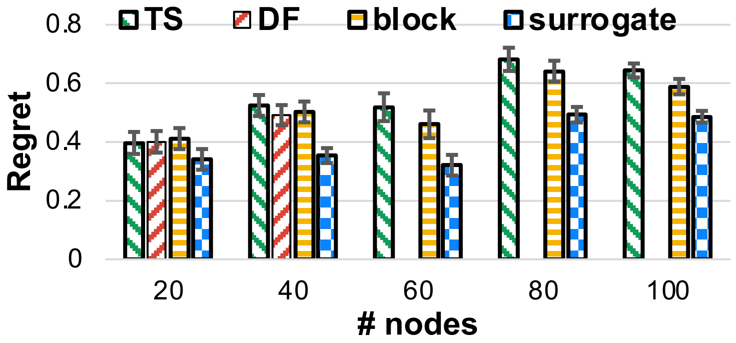
Experimental Results

Convex/non-convex optimization

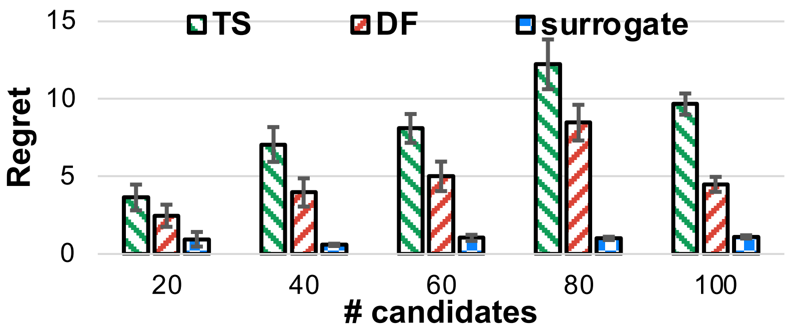
AAMAS 2020
NeurIPS 2020
spotlight

	Two-stage	DF	Block	Surrogate
scalability	-	Poor	Better	Best
performance	Poor	Good	Good (guarantee)	Best

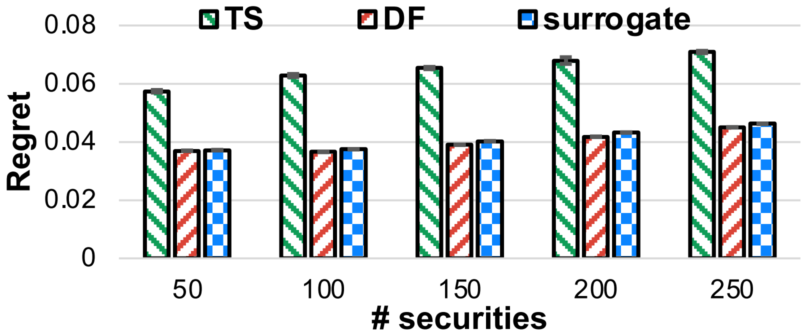
Smuggling intervention



Movie recommendation



Portfolio optimization



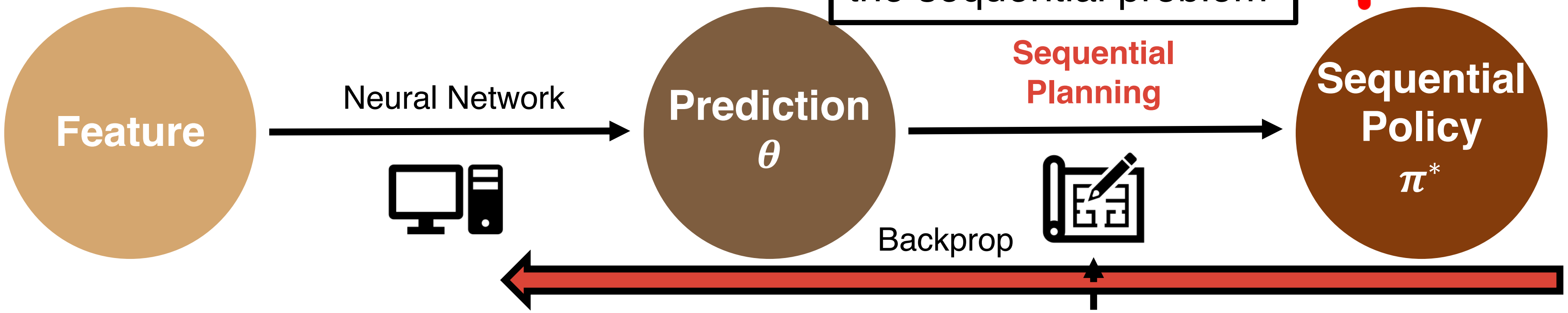
Outline

- **Decision-focused Learning (Recap)**
- **Scalability and Applications in Different Optimization Problems**
  - Convex/non-convex optimization
  - Sequential optimization
  - Multi-agent optimization
- **Summary of Differentiation Techniques**

Decision-focused Learning (Recap)

Sequential planning

Forward path: solving the sequential problem



Backward path: differentiating through the planning problem

$$\frac{d\pi^*}{d\theta} = \left(\nabla_\pi^2 J_\theta(\pi^*)\right)^{-1} \nabla_{\theta\pi}^2 J_\theta(\pi^*)$$

- Policy optimality or Bellman optimality
- Policy gradient theorem

Expensive computation!!

Public Health Challenges

Sequential planning

Forward path: solving the sequential problem

Feature → Neural Network → Prediction θ → Sequential Planning → Sequential Policy π^*

Backprop

Backward path: differentiating through the planning problem

armman

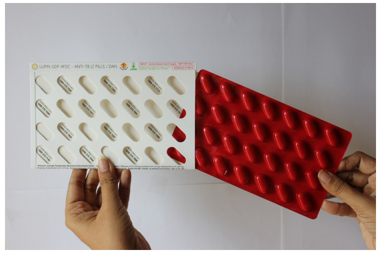
Schedule service calls to provide health information to pregnant women

2019-2020 Women and children enrolled **264,208**

As of March 2020 Sakhis Trained **7349**

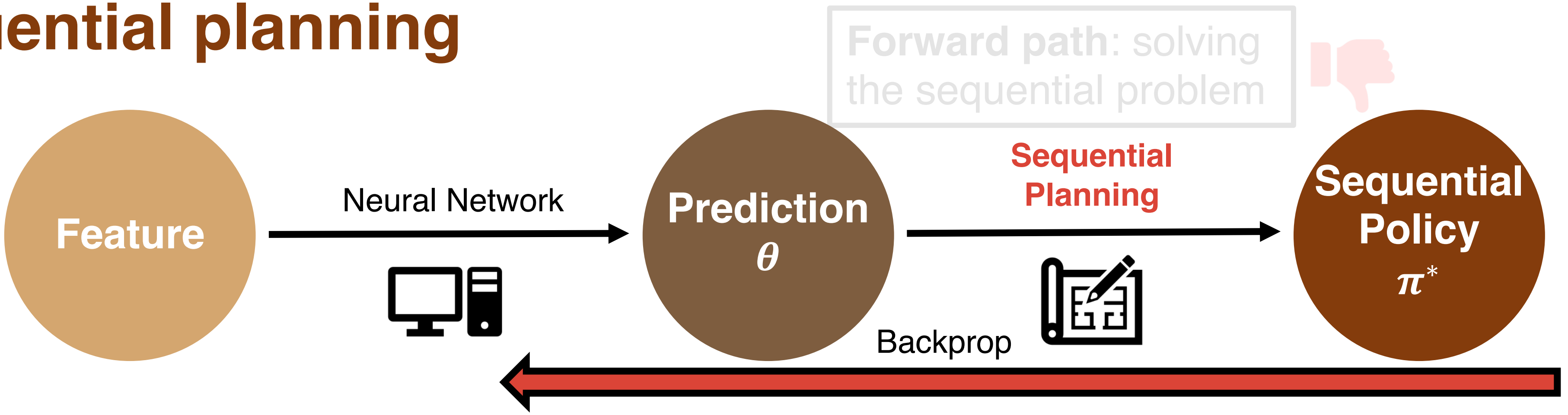
99DOTS

Schedules service calls to improve adherence of tuberculosis patients



Public Health Challenges

Sequential planning



Backward path: differentiating through

Can decision-focused learning be applied to **large-scale sequential** problems?

to pregnant women

tuberculosis patients

2019-2020

Women and children enrolled

264,208

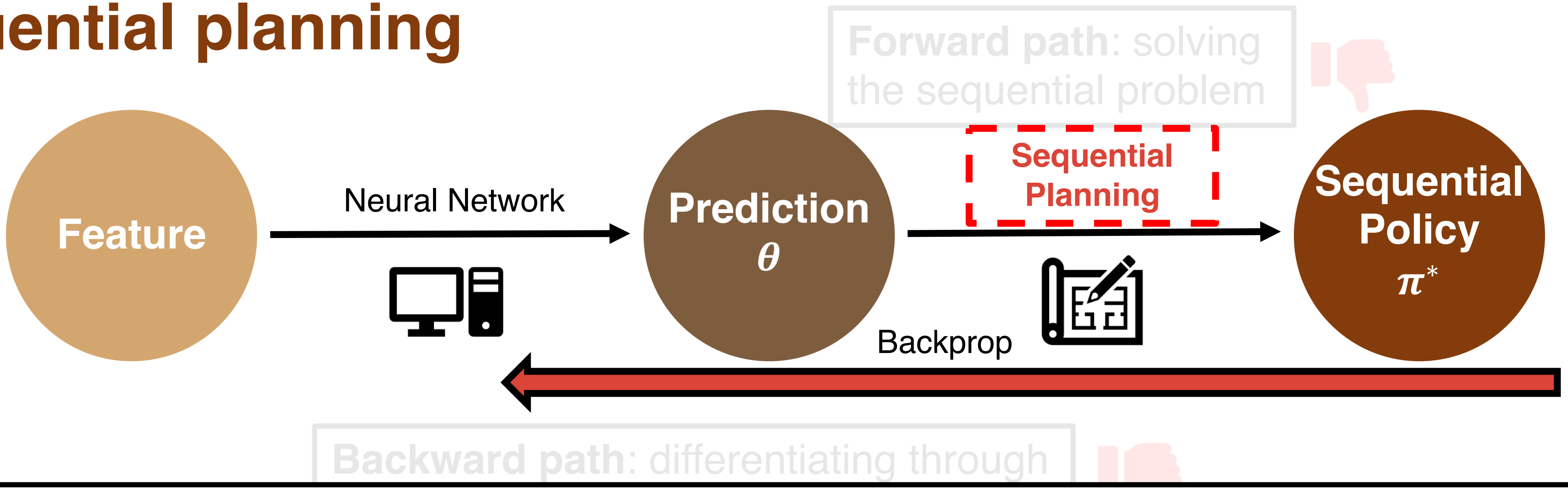
As of March 2020

Sakhis Trained

7349

Public Health Challenges

Sequential planning

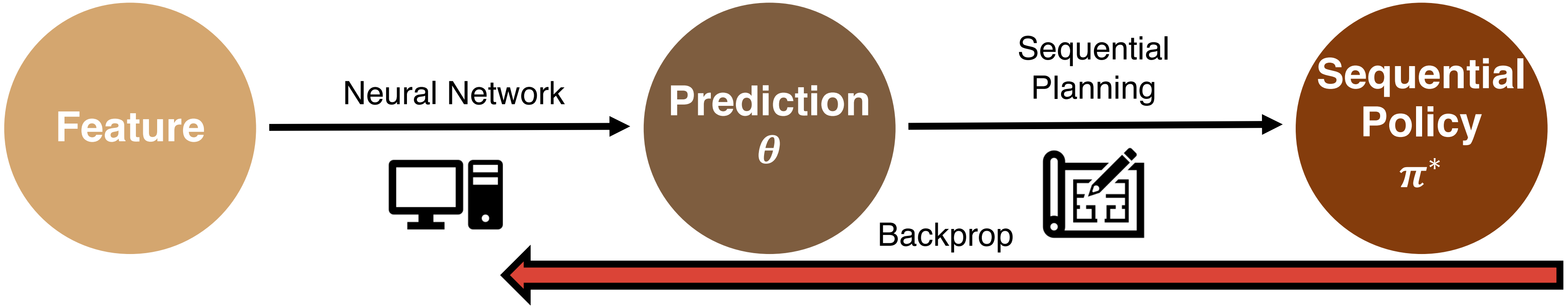


Can decision-focused learning be applied to **large-scale sequential** problems?

Key idea: leverage property of the sequential problems

Maternal and Child Health

Sequential planning

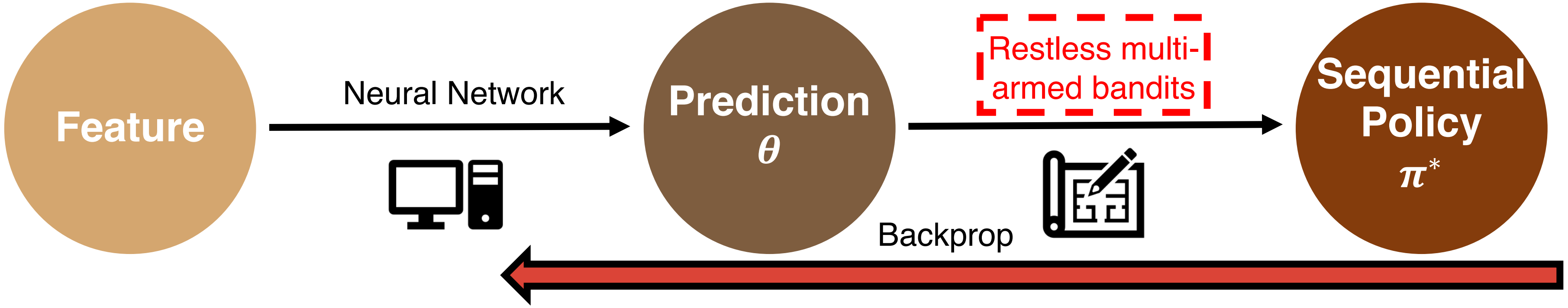


Predictive problem:
- Participants' transition probabilities

Sequential planning (RMAB):
- Schedule service calls based on observed states

Maternal and Child Health

Sequential planning



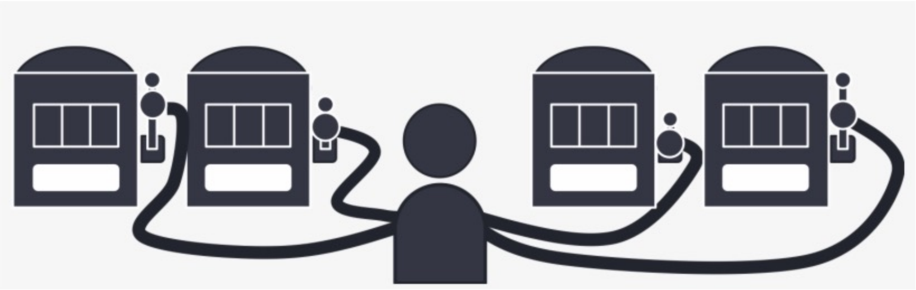
Predictive problem:

- Participants' transition probabilities

Sequential planning (RMAB):

- Schedule service calls based on observed states

Sequential Problem: Restless Bandits



- An extension of multi-armed bandits
- **Restless state**: arms are associated with states and transition functions
- **Action**: select K out of N arms to pull
- **Goal**: maximize total reward in T steps

$s_1 \quad s_2 \quad \cdots \quad s_N \in S$

$a_1 \quad a_2 \quad \cdots \quad a_N \in A = \{0,1\}$

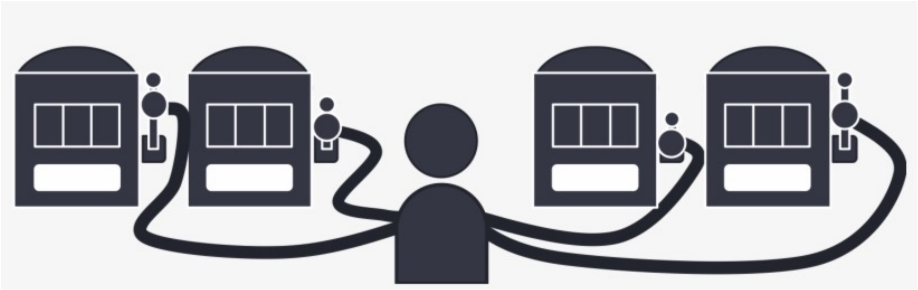
$P_1 \quad P_2 \quad \cdots \quad P_N \in S \times A \times S \rightarrow \mathbb{R}$

$s_1' \quad s_2' \quad \cdots \quad s_N' \in S$

PSPACE-hard to find the optimal solution!

Sequential Problem: Restless Bandits

- An extension of multi-armed bandits



$s_1 \quad s_2 \quad \cdots \quad s_N \in S$

- **Restless state**: arms are associated with states and transition functions

Pull the largest K

$W_1(s_1) \; W_2(s_2) \quad \cdots \quad W_N(s_N)$

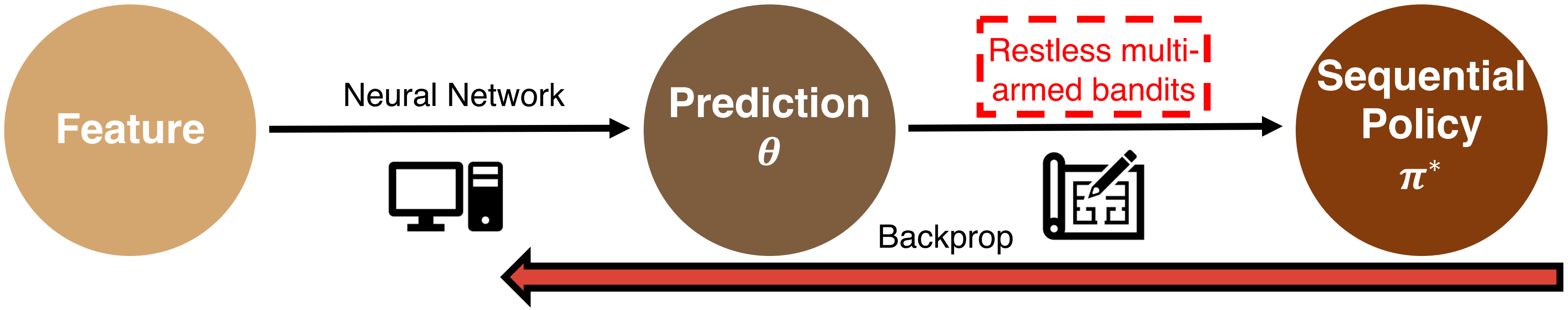
- **Whittle index**: the value of pulling

$W_i(s_i) :=$ smallest subsidy m provided to not pulling $(a = 0)$

s.t. $\quad Q_i(s_i, a = 0) = Q_i(s_i, a = 1)$

Maternal and Child Health

Sequential planning



Predictive problem:

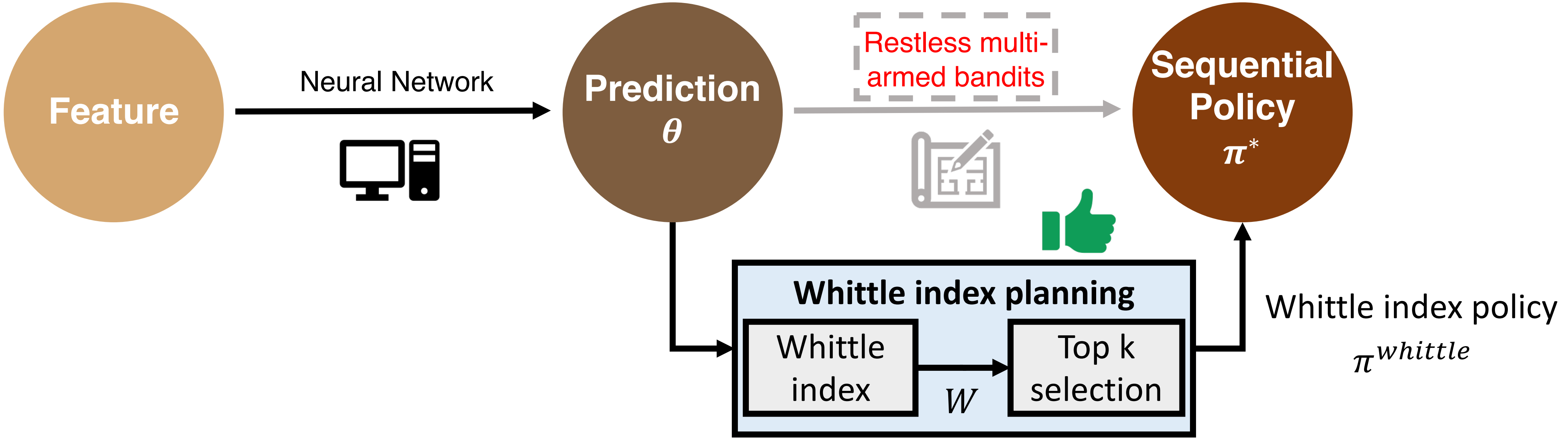
- Participants' transition probabilities

Sequential planning (RMAB):

- Schedule service calls based on observed states

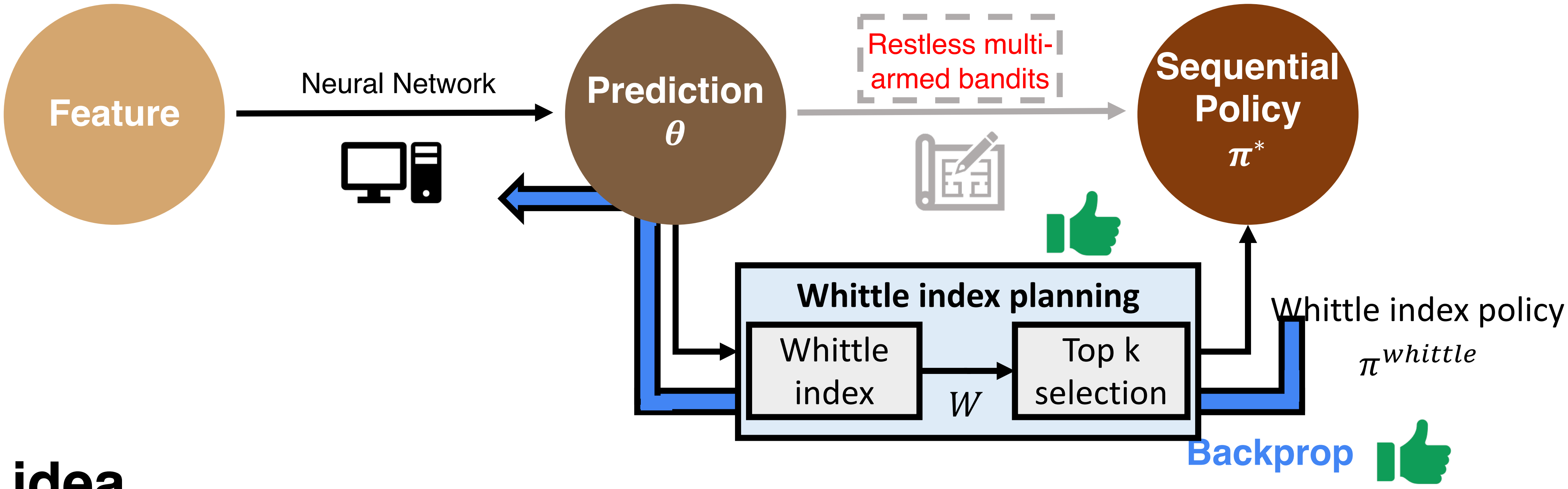
Maternal and Child Health

Sequential planning



Whittle Index Differentiability

Sequential planning



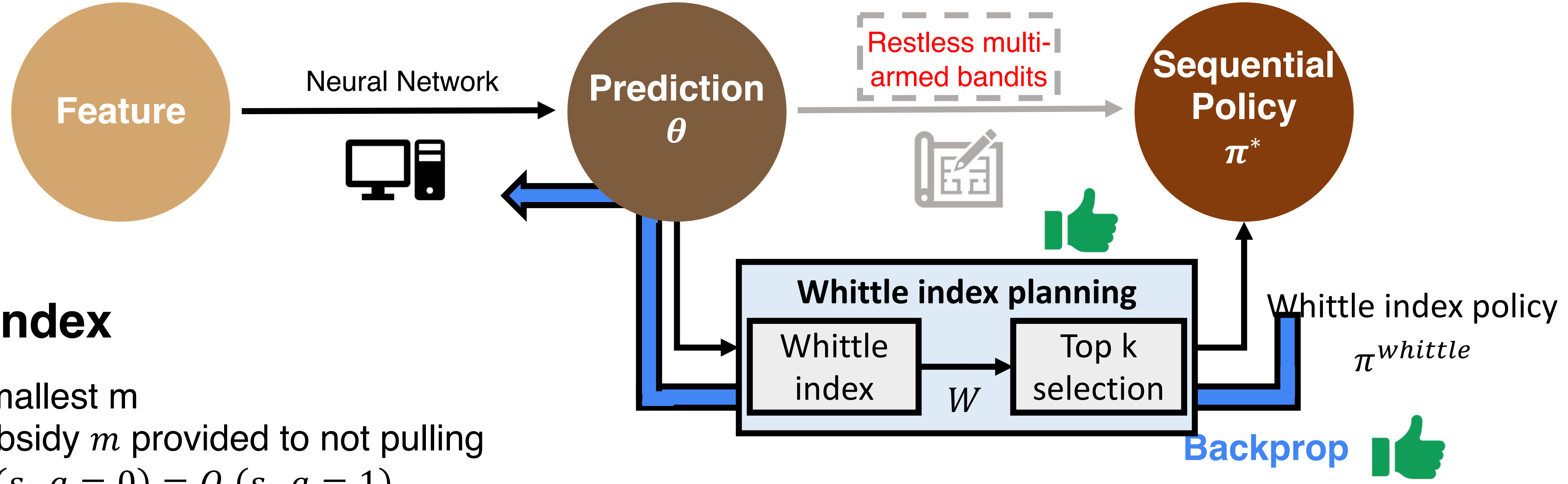
Main idea

- Differentiate through Whittle index policy

  Whittle index

  Top-k selection

Whittle Index Differentiability
Sequential planning



Whittle index

$W_i(s_i) :=$ smallest m

s.t. subsidy m provided to not pulling

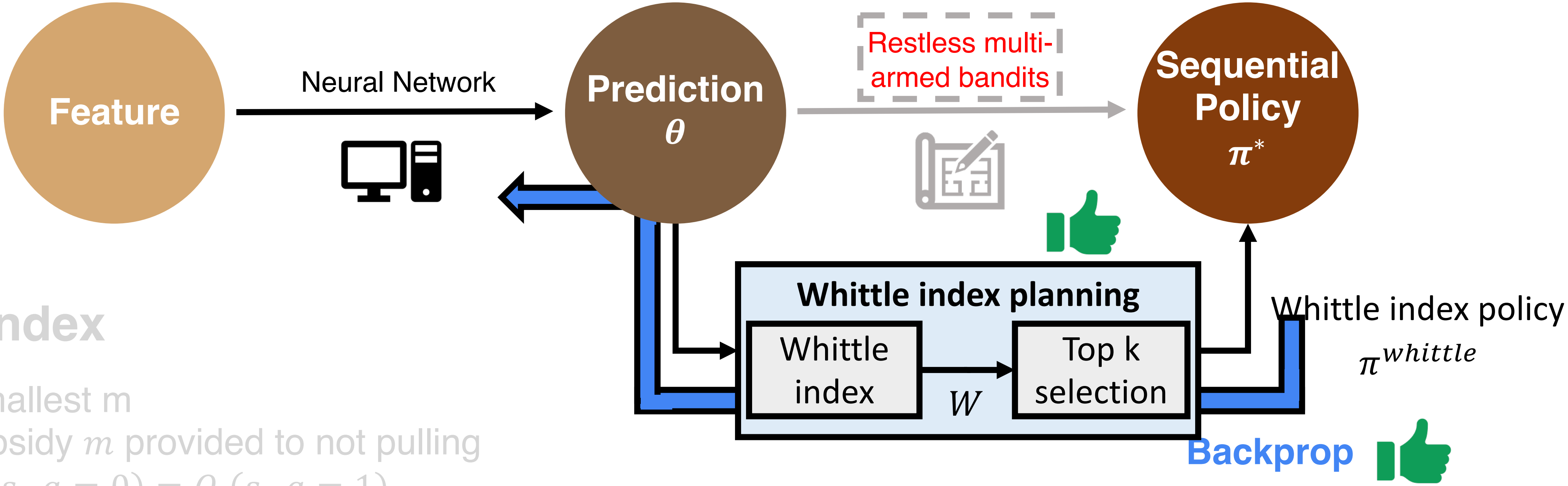
$Q_i(s_i, a = 0) = Q_i(s_i, a = 1)$

Bellman equation parameterized by θ

Top-k selection Amos et al. (arXiv 2019), Xie et al. (NeurIPS 2020)

Whittle Index Differentiability

Sequential planning



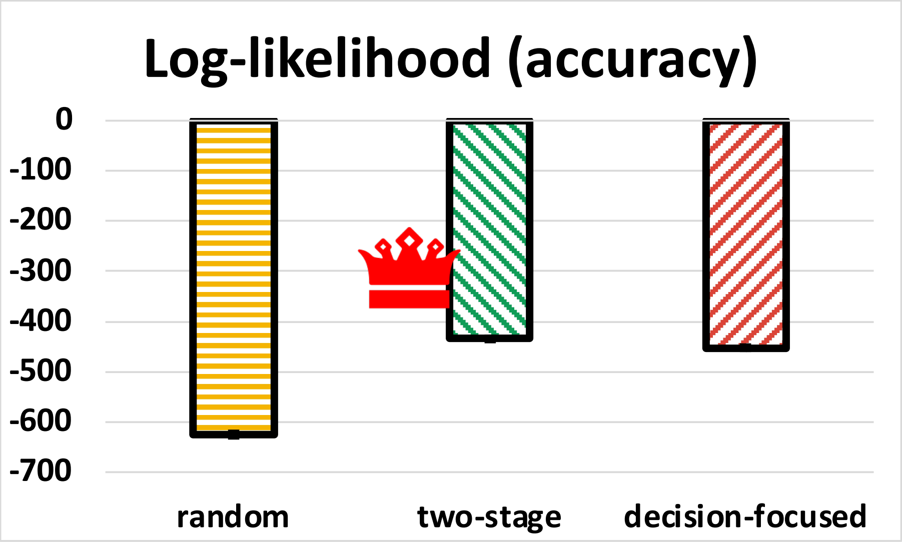
Whittle index

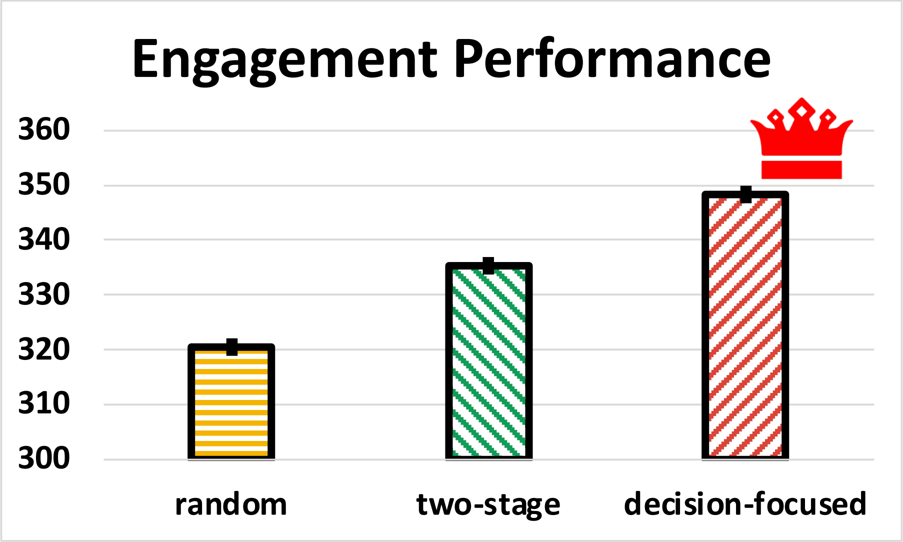
$W_i(s_i) :=$ smallest m
s.t. subsidy m provided to not pulling
$Q_i(s_i, a = 0) = Q_i(s_i, a = 1)$

Takeaway: leverage approximate solution to bypass the cost of differentiating through sequential problems

Maternal and Child Health
Sequential planning

Scale up decision-focused learning to large sequential problem





Collaboration

AI in healthcare

Outline

- **Decision-focused Learning (Recap)**
- **Scalability and Applications in Different Optimization Problems**
  - Convex/non-convex optimization
  - Sequential optimization
  - Multi-agent optimization
- **Summary of Differentiable Optimization**

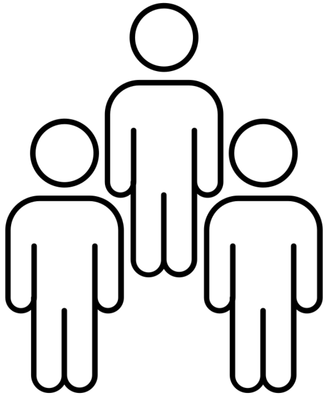
Stackelberg Games with Multiple Followers

AAAI 2022

Multi-agent planning

Leader

- Choose a strategy $\pi \in \Pi$ first
- Receive a payoff $f(z^*, \pi)$ and a constraint cost $g(z^*, \pi)$



n followers

- Followers select strategies simultaneously to form an equilibrium $z^* = [z_1, z_2, \ldots, z_n]$
- Follower i receives $f_i(z^*, \pi)$

Applications

Wildlife conservation

Cybersecurity

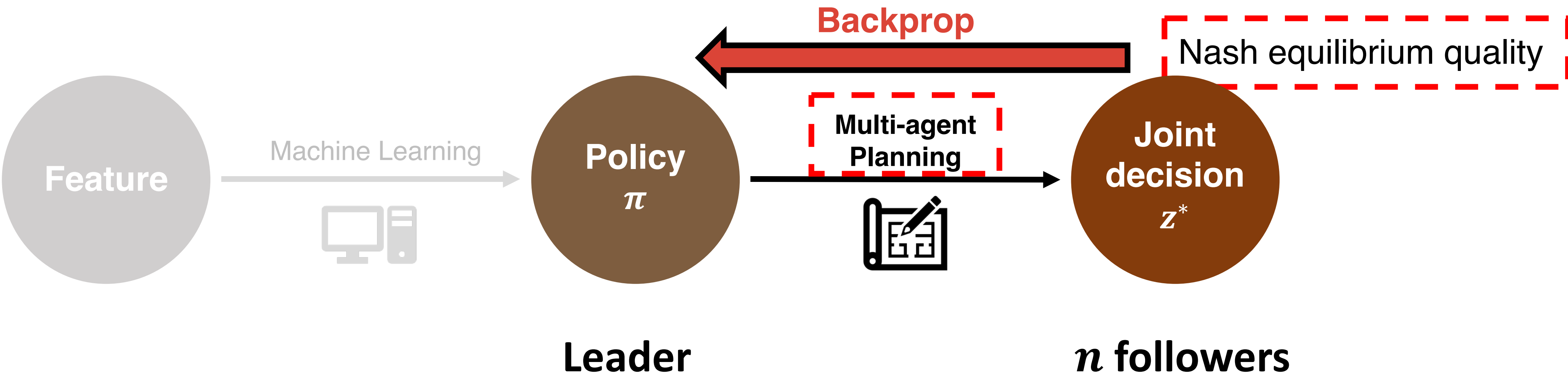
Public health

Public safety

Stackelberg Games with Multiple Followers

AAAI 2022

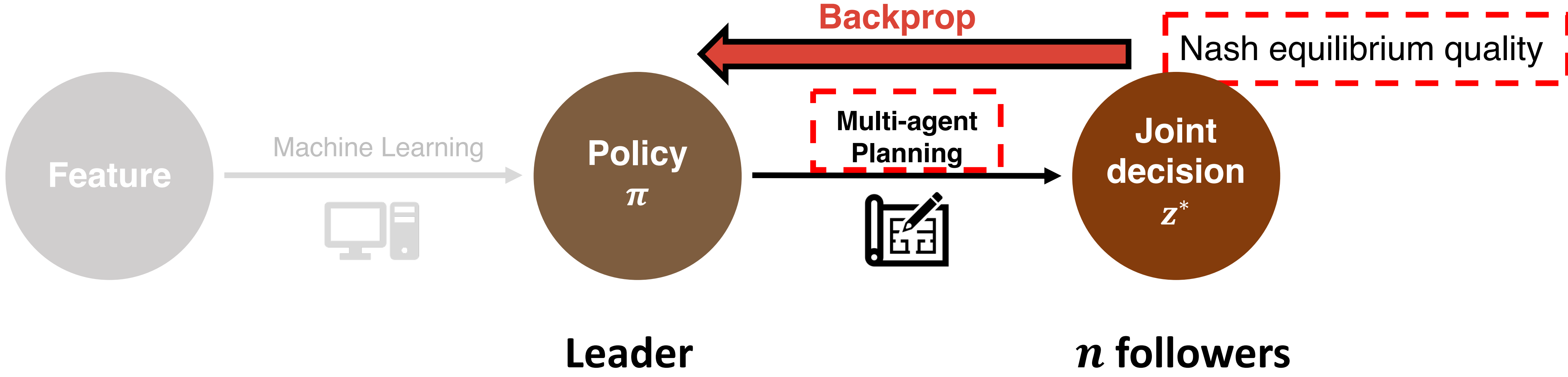
Multi-agent planning



Stackelberg Games with Multiple Followers

AAAI 2022

Multi-agent planning



Algorithm

- Run gradient descent to optimize Nash equilibrium quality

Main idea

- Differentiability of Nash equilibrium and multi-agent planning
- Concatenate all the KKT conditions: $KKTs = [KKT_1, KKT_2, \ldots, KKT_n]$

Stackelberg Games with Multiple Followers

AAAI 2022

Multi-agent planning

Bilevel optimization

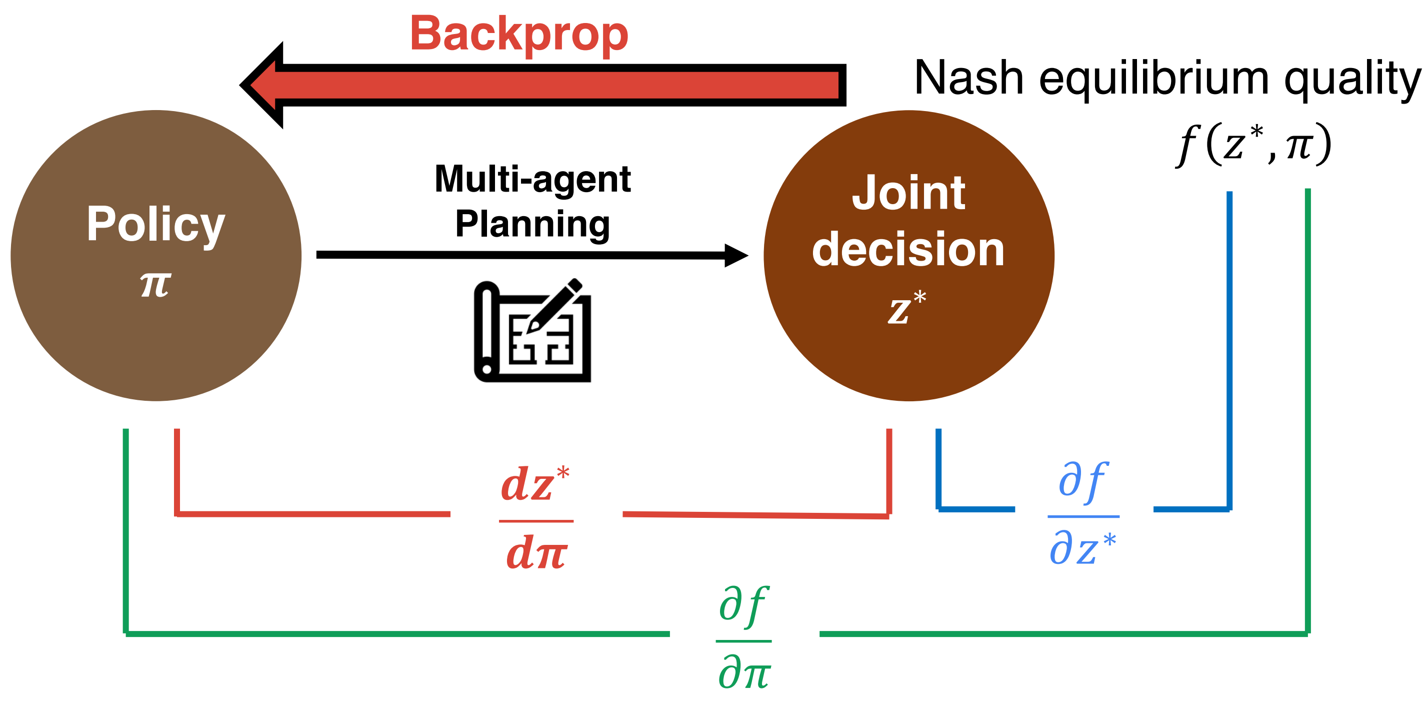
$$\max_{\pi \in \Pi} f(z^*, \pi)$$
$$s.t. \;\; z^* = \mathcal{O}(\pi), \;\; g(z^*, \pi) \leq 0$$

where $\mathcal{O}: \Pi \rightarrow Z^*$ is an oracle that returns a Nash equilibrium

Gradient-based algorithm

- **Challenge:** gradient computation

$$\frac{df(z^*, \pi)}{d\pi} = \frac{\partial f}{\partial \pi} + \frac{\partial f}{\partial z^*} \frac{dz^*}{d\pi}$$



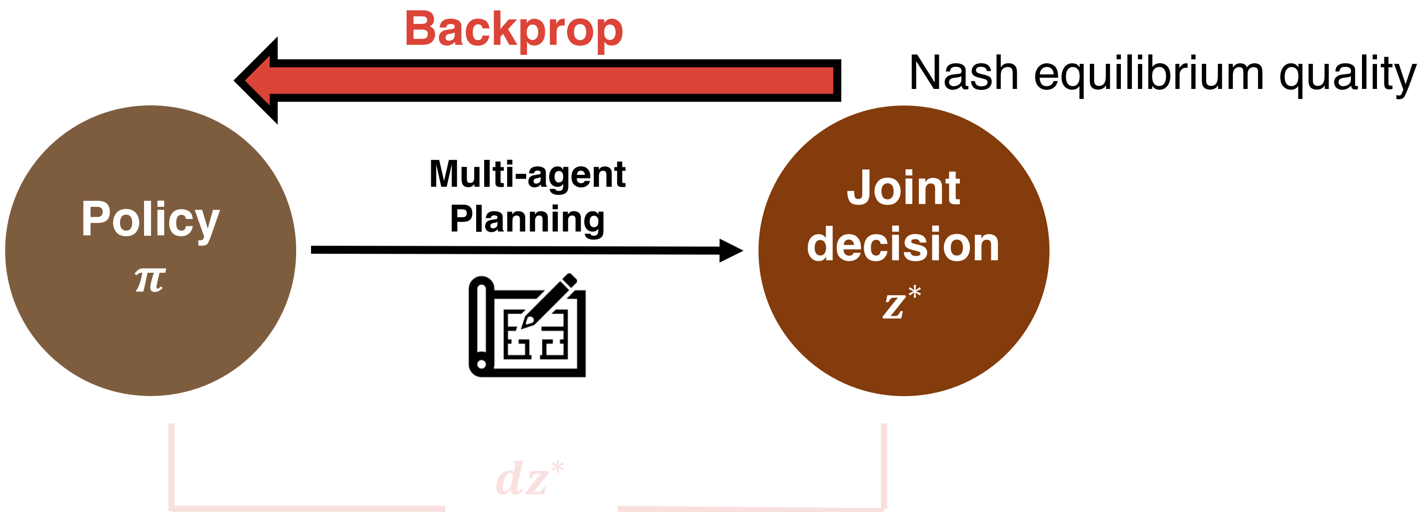
Stackelberg Games with Multiple Followers

AAAI 2022

Multi-agent planning

Gradient-based algorithm

$$\frac{df(z^*,\pi)}{d\pi} = \frac{\partial f}{\partial \pi} + \frac{\partial f}{\partial z^*}\frac{dz^*}{d\pi}$$



Takeaway: differentiability of Nash equilibria and its application to Stackelberg games

$$KKTs(z^*,\pi) = 0 \quad \Rightarrow \frac{\partial KKTs}{\partial \pi} + \frac{\partial KKTs}{\partial z}\frac{dz^*}{d\pi} = 0 \quad \Rightarrow \frac{dz^*}{d\pi} = \left(\frac{\partial KKTs}{\partial z}\right)^{-1}\frac{\partial KKTs}{\partial \pi}$$

Outline

- **Decision-focused Learning (Recap)**
- **Scalability and Applications in Different Optimization Problems**
  - Convex/non-convex optimization
  - Sequential optimization
  - Multi-agent optimization
- **Summary of Differentiable Optimization**

Summary

Differentiable optimization is a powerful primitive to embed **non-trivial modeling knowledge** to use within larger systems

- **Theory** and **engineering** foundation
- **Scalability** in larger systems
- **Extension** to more optimization problems

Scalability Challenges and Solutions to Differentiable Optimization

Kai Wang • Harvard University

Differentiable Surrogate: Automatically Learning Compact Surrogates [NeurIPS 2020]
Differentiable Block Sampling: Scalable Game-focused Learning [AAMAS 2020]
Differentiable RL: Learning MDPs from Features [NeurIPS 2021]
Differentiable Whittle Index: DFL in Restless Bandits [arXiv 2022]
Differentiable Equilibria: Coordinating Followers to Reach Better Equilibria [AAAI 2022]

Joint work with Bryan Wilder, Sanket Shah, Lily Xu, Aditya Mate, Haipeng Chen, Andrew Perrault, Aparna Taneja, Michael K Reiter, Finale Doshi-Velez, Milind Tambe

Kai Wang (kaiwang@g.harvard.edu)
Andrew Perrault (perrault.17@osu.edu)
Brandon Amos (bda@meta.com)