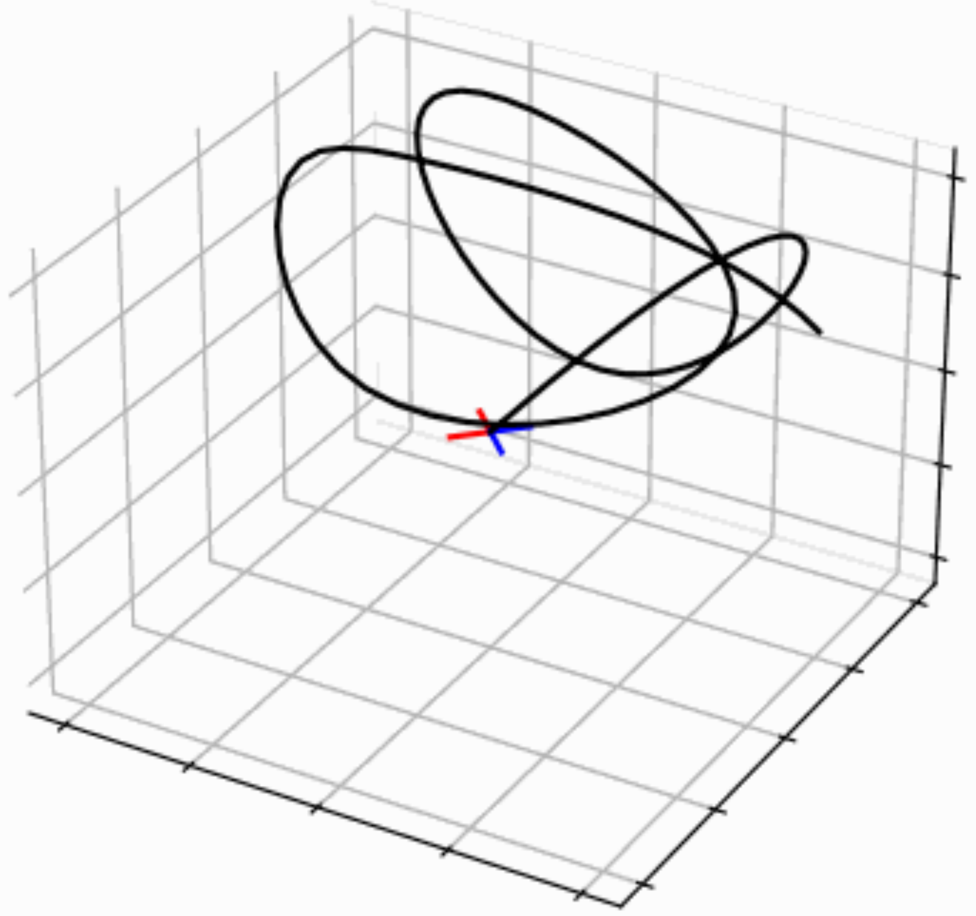
Learning to Accelerate Optimizers with Guarantees

Harvard Computational Robotics Talk 2024
Rajiv Sambharya

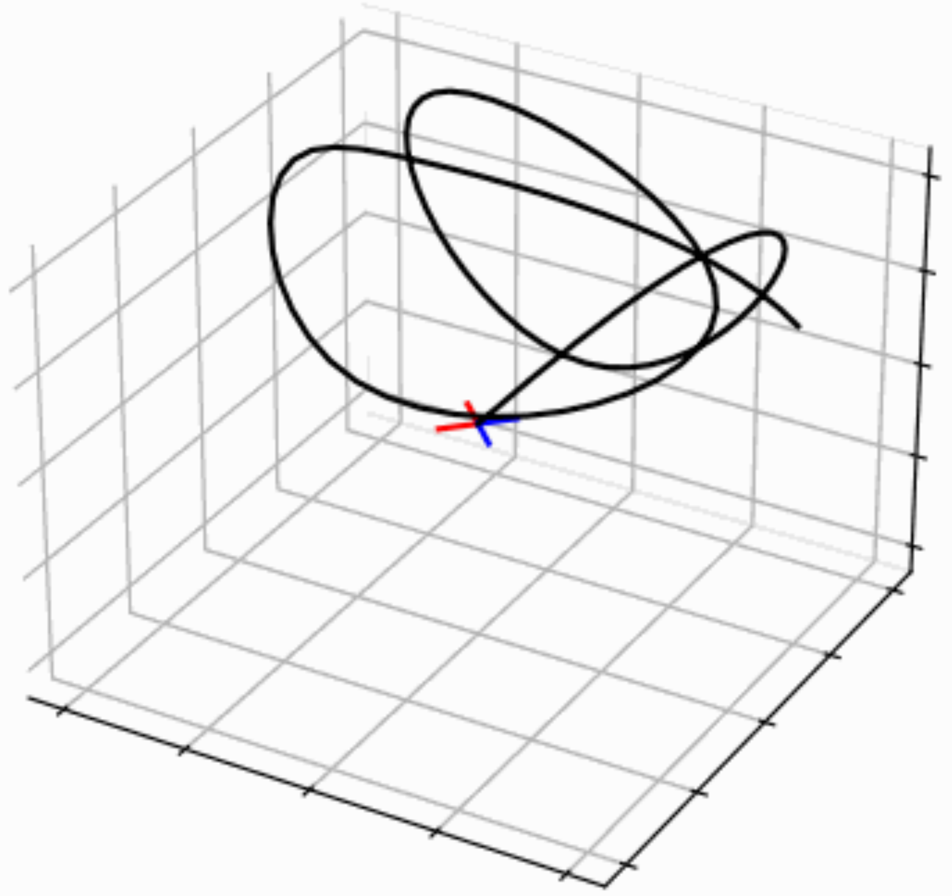
PRINCETON UNIVERSITY

ORFE

Tracking a reference trajectory with a quadcopter



Success!
(If given enough time)



Failure: not enough time to solve

Model predictive control

optimize over a smaller horizon (T steps),
implement first control,
repeat

Current state,
reference trajectory →

Model predictive controller

minimize $\sum_{t=1}^{T} \|x_t - x_t^{\text{ref}}\|_2^2$

subject to $x_{t+1} = Ax_t + Bu_t$

$x_t \in \mathcal{X}, \quad u_t \in \mathcal{U}$

$x_0 = x_{\text{init}}$

→ Control
inputs

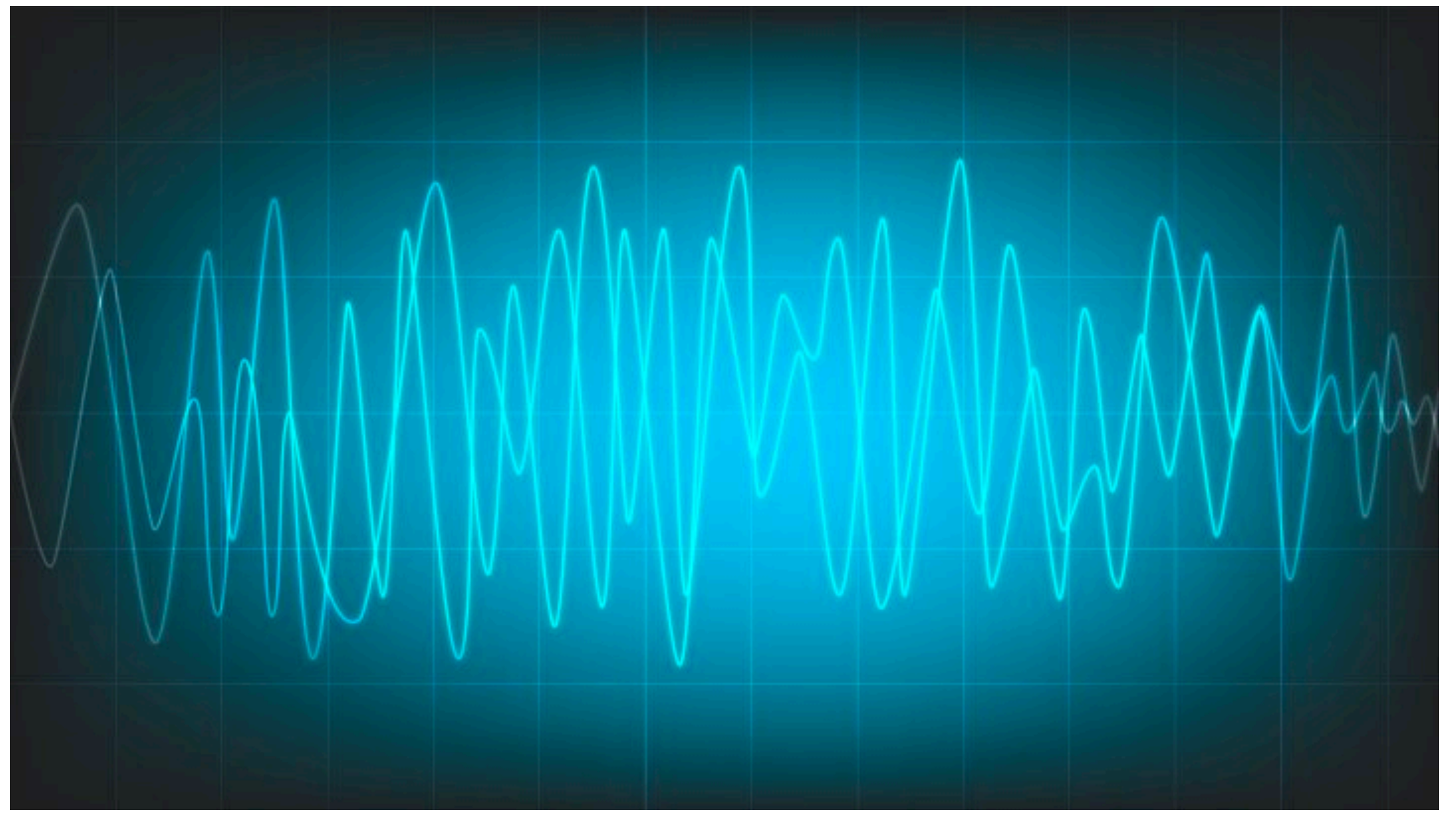
Challenge: we need faster methods for optimization

Claim: real-world optimization is parametric

Robotics and control

Energy

Signal processing



Can machine learning speed up parametric optimization?

Goal: Do mapping quickly and accurately

Parameter

Optimal solution

$$\theta \longrightarrow \begin{array}{ll} \text{minimize} & f_\theta(z) \\ \text{subject to} & g_\theta(z) \leq 0 \end{array} \longrightarrow z^\star(\theta)$$

$\theta \longrightarrow$ Only Optimization $\longrightarrow \hat{z}^{\mathrm{Opt}}(\theta)$

$\theta \longrightarrow$ Only Machine Learning $\longrightarrow \hat{z}^{\mathrm{ML}}(\theta)$

$\theta \longrightarrow$ Optimization Machine Learning $\longrightarrow \hat{z}^{\mathrm{Opt/ML}}(\theta)$

Learning to Optimize

The learning to optimize paradigm

Goal: solve the parametric optimization problem fast

$$\begin{array}{ll} \text{minimize} & f_\theta(z) \\ \text{subject to} & g_\theta(z) \leq 0 \end{array}$$

Offline

Data collection

Parameters θ

Solve

Optimal solutions $z^\star$

Training

Training parameter θ

Learnable Optimizer with weights w

Candidate solution $\hat{z}_w(\theta)$

Loss

Learn

Deploy

Online evaluation

Unseen parameter θ

Learned Optimizer

High-quality solution

Challenges in learning to optimize methods

- **I: Lack convergence guarantees**
- **II: Lack generalization guarantees**
- **III: Hard to integrate with state-of-the-art solvers**

We need reliable L2O methods

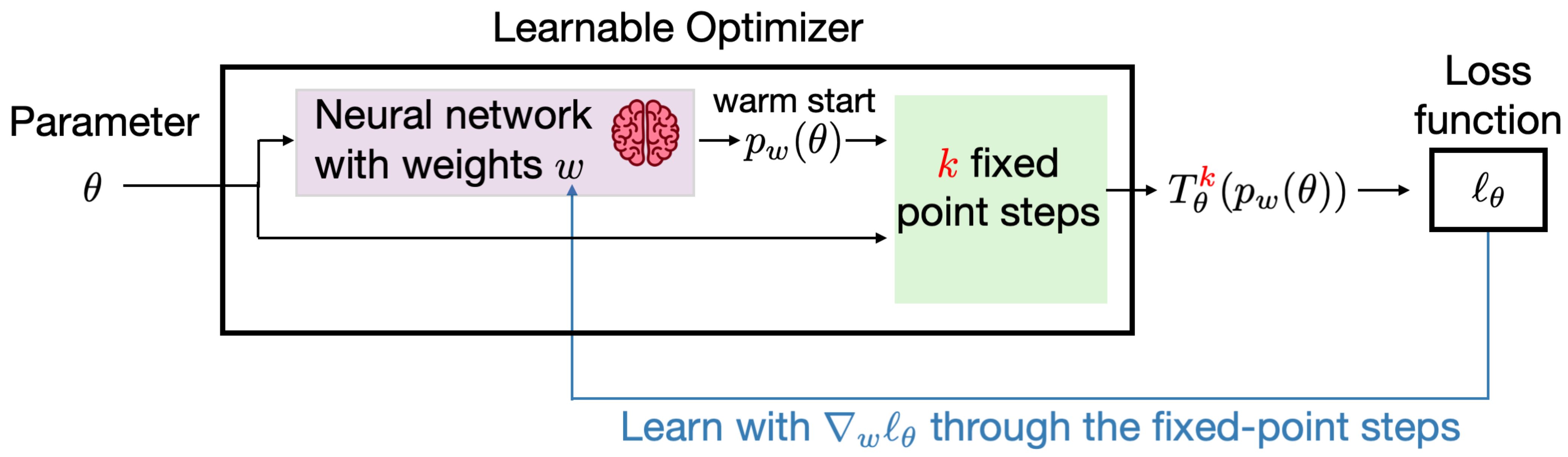
Learning to Optimize: A Primer and A Benchmark **[Chen. et al 2021]**

"So, to conclude this article, let us quote Sir Winston Churchill: 'Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.'"

Talk Outline

- **Part 1: Learning to Warm-Start Fixed-Point Optimization Algorithms**

- **Part 2: Practical Performance Guarantees for Classical and Learned Optimizers**



Learnable Optimizer

Parameter

θ

Neural network with weights w

warm start $p_w(\theta)$

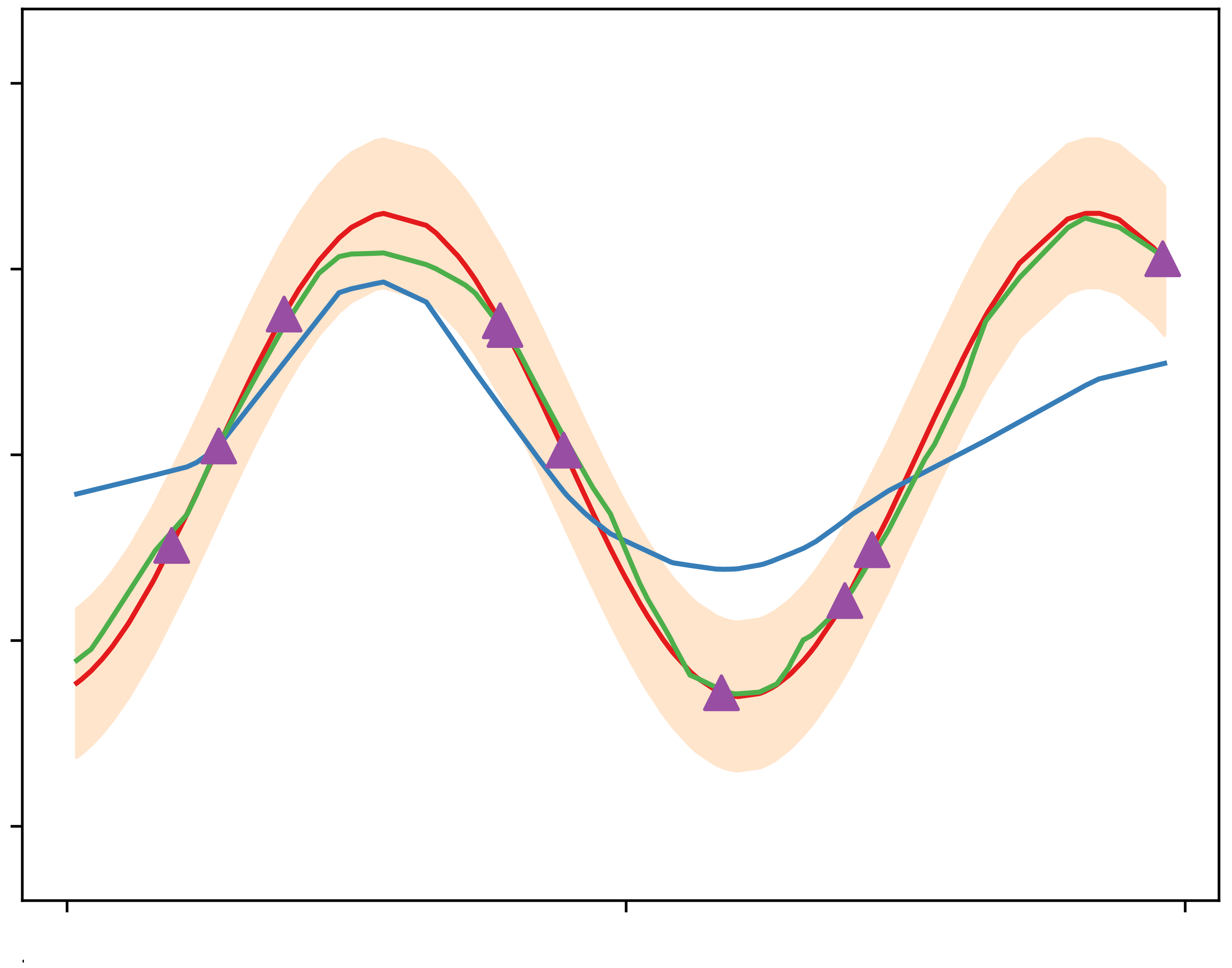
k fixed point steps

$T_\theta^k(p_w(\theta))$

Loss function

ℓ_θ

Learn with $\nabla_w \ell_\theta$ through the fixed-point steps



Collaborators

Georgina
Hall

INSEAD

Brandon
Amos

Meta

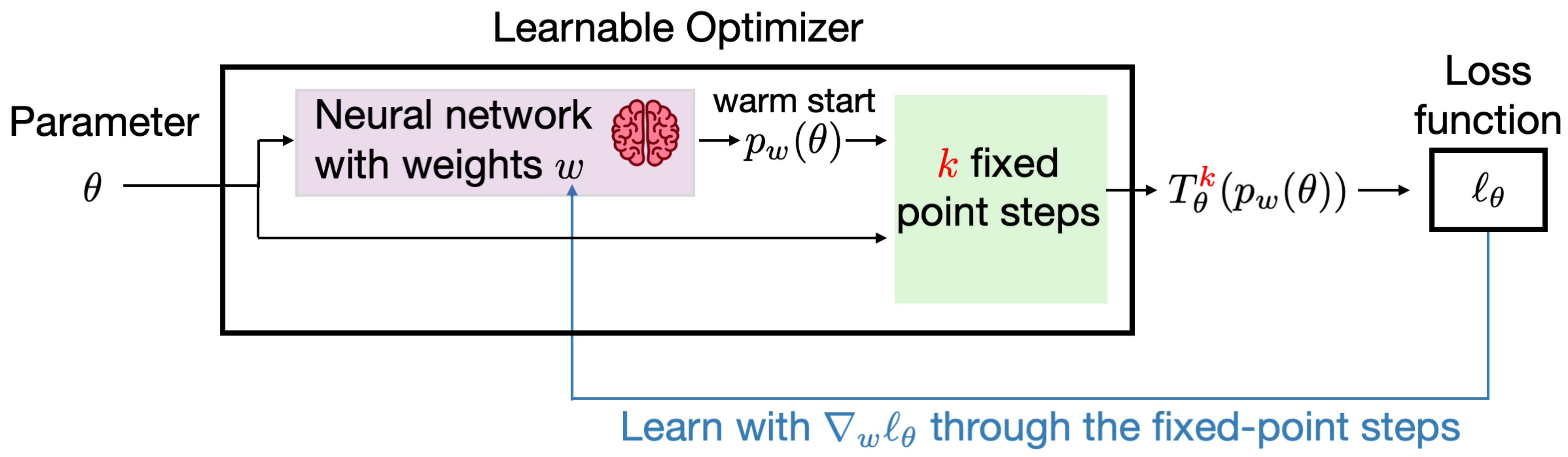
Bartolomeo
Stellato

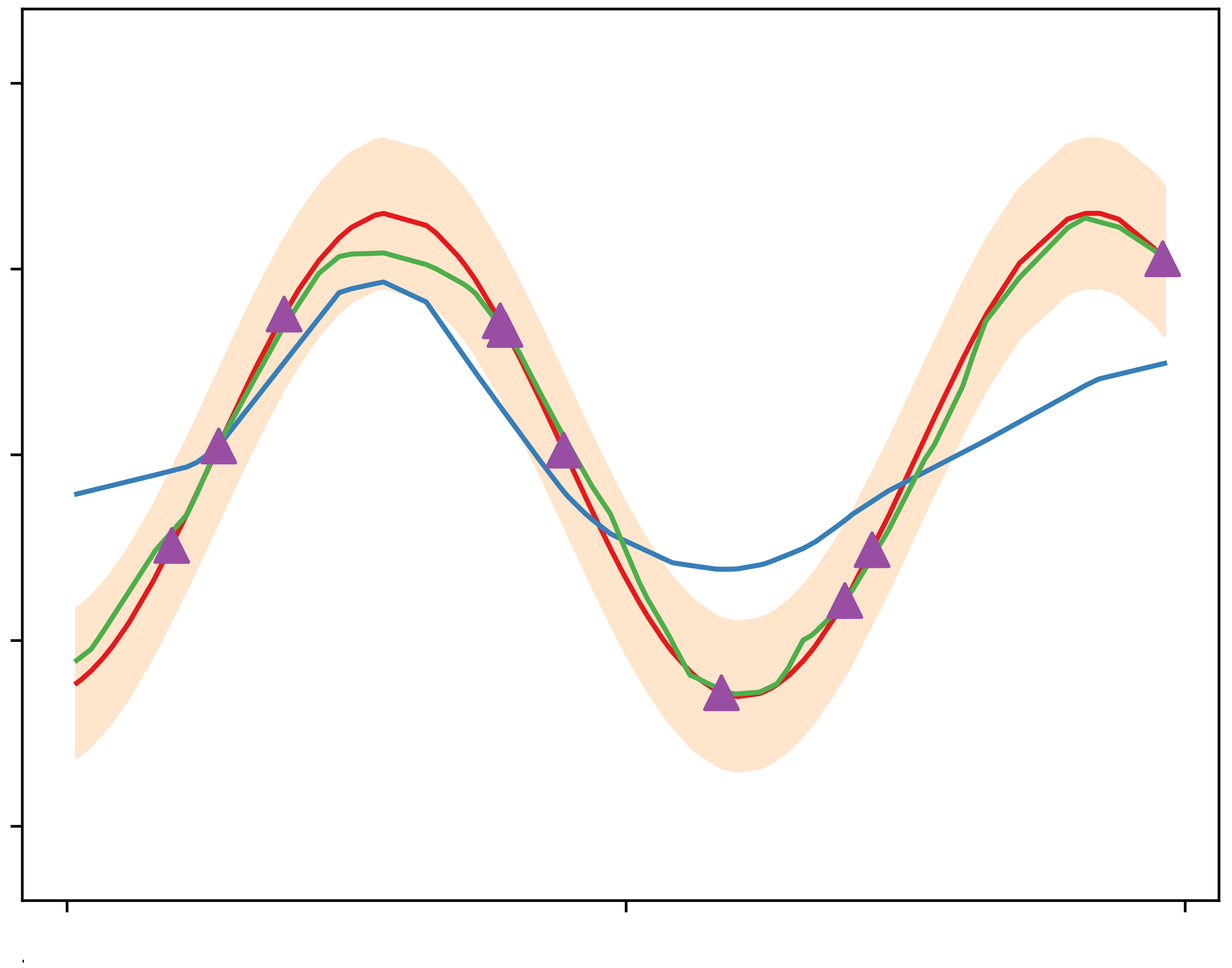
PRINCETON
UNIVERSITY

Talk Outline

- **Part 1: Learning to Warm-Start Fixed-Point Optimization Algorithms**

- **Part 2: Practical Performance Guarantees for Classical and Learned Optimizers**





Fixed-point optimization problems are ubiquitous

Parametric fixed-point problem: find z such that $z = T_\theta(z)$

Convex optimization

Problem
minimize $f_\theta(z)$
subject to $g_\theta(z) \leq 0$

→

Optimality conditions
(KKT conditions)

→

Fixed-point operator

Unconstrained, smooth convex optimization

Problem
minimize $f_\theta(z)$

Smooth, convex

→

Optimality conditions
$\nabla f_\theta(z) = 0$

→

Fixed-point operator
$T_\theta(z) = z - \alpha \nabla f_\theta(z)$

Many optimization algorithms are fixed-point iterations

Fixed-point iterations: $z^{i+1} = T_\theta(z^i)$

Initialize with z^0 (a warm-start)

Terminate when $\|T_\theta(z^j) - z^j\|_2$ is small

Fixed-point residual

Example: Proximal gradient descent

$$\text{minimize} \quad g_\theta(z) + h_\theta(z)$$

Convex
Smooth

Convex
Non-smooth

Iterates $z^{i+1} = \textbf{prox}_{\alpha h_\theta}(z^i - \alpha \nabla g_\theta(z^i))$

$$\textbf{prox}_s(v) = \arg\min_x \left(s(x) + \frac{1}{2}\|x - v\|_2^2 \right)$$

Problem: limited iteration budget

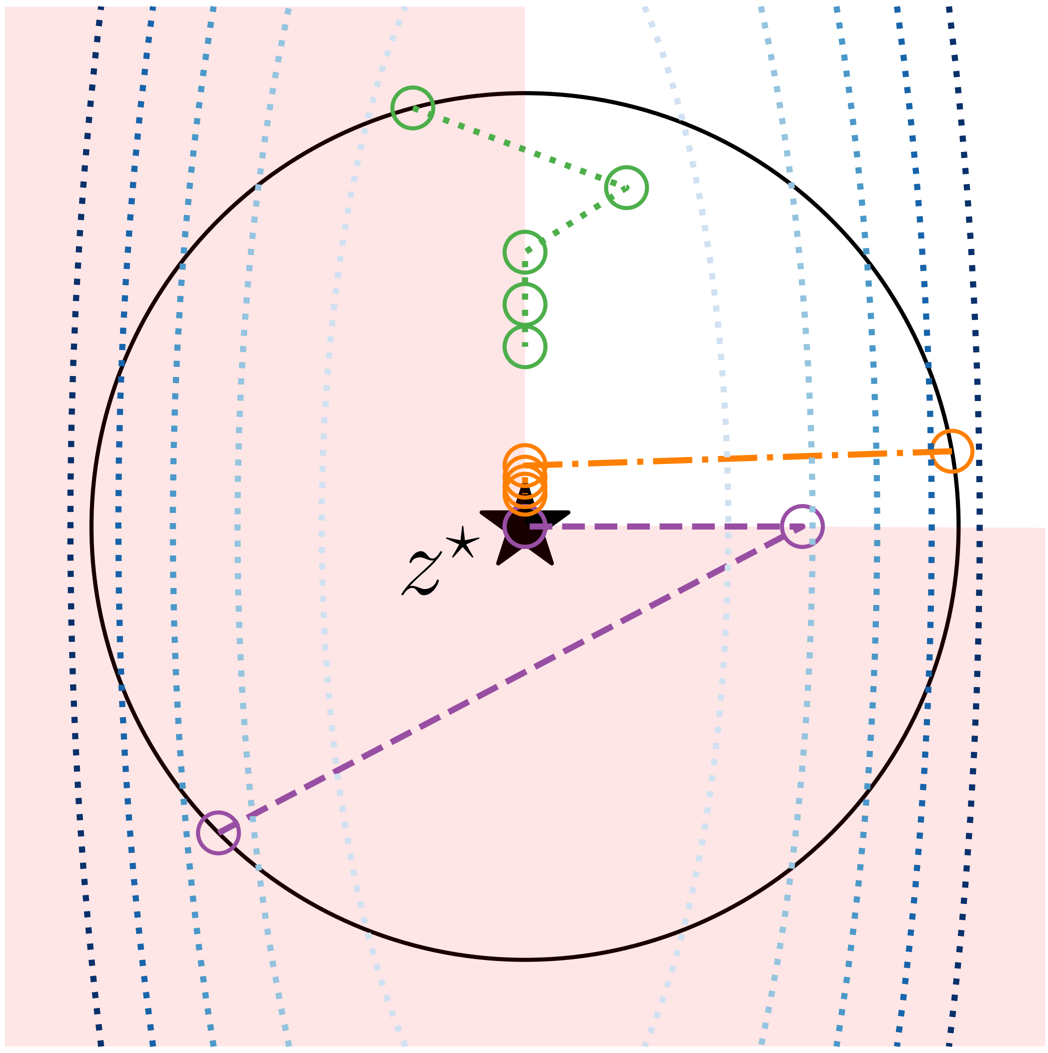
Solution: learn the warm-start to improve the solution within budget

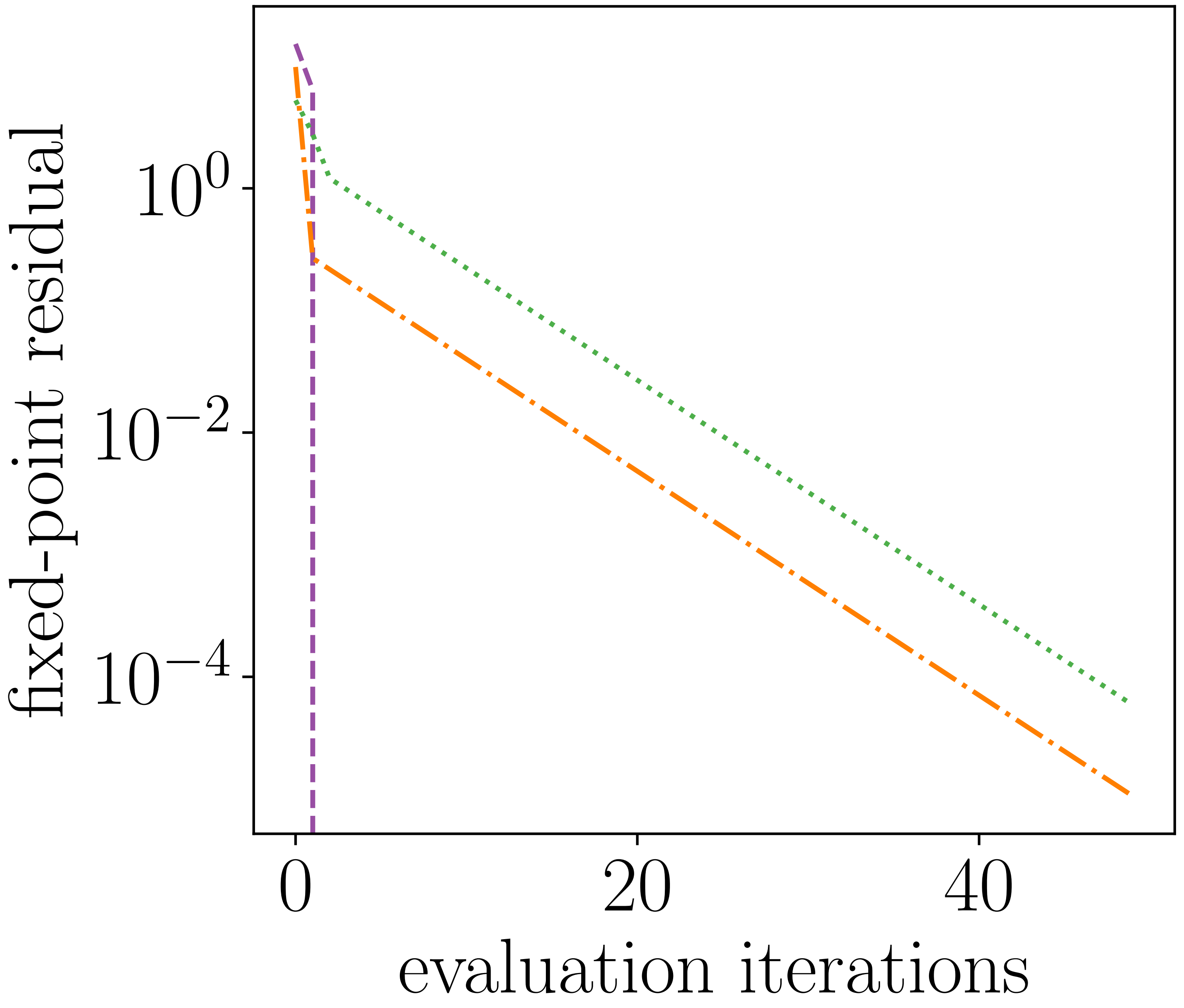
Some warm starts are better than others

$$\begin{array}{ll} \text{minimize} & 10z_1^2 + z_2^2 \\ \text{subject to} & z \geq 0 \end{array}$$

★ Optimal solution at the origin

Run proximal gradient descent to solve

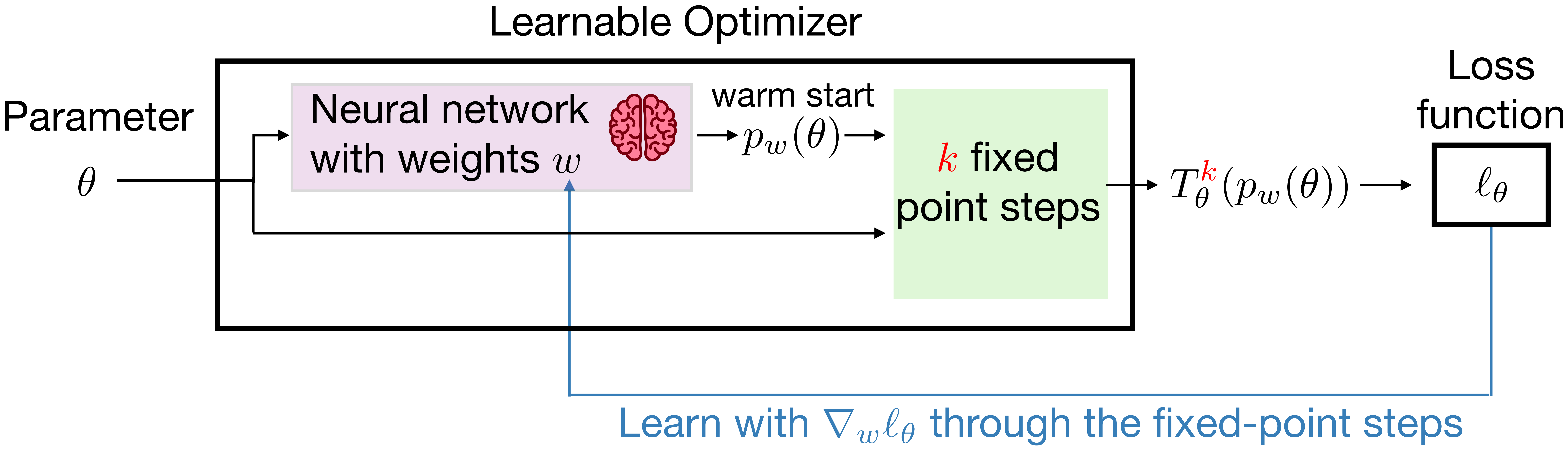




All three warm starts appear to be equally suboptimal but converge at very different rates

The quality of the warm start depends on the algorithm

End-to-end learning architecture



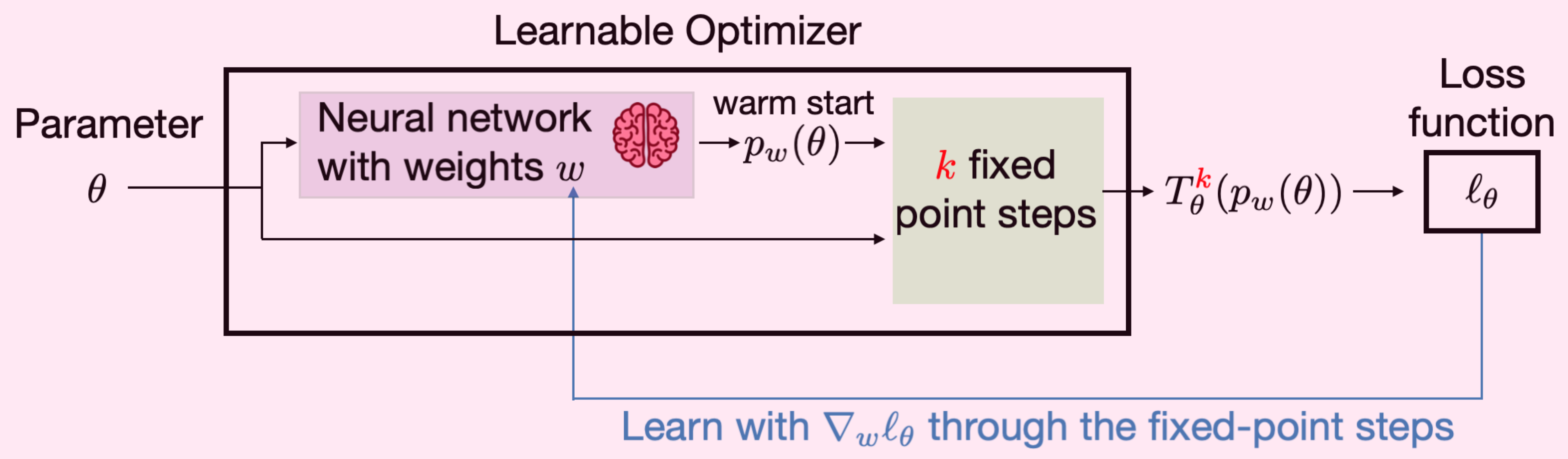
Learn with $\nabla_w \ell_\theta$ through the fixed-point steps

Loss function: $\ell_\theta(z) = \|z - z^\star(\theta)\|_2^2$ Ground truth solution

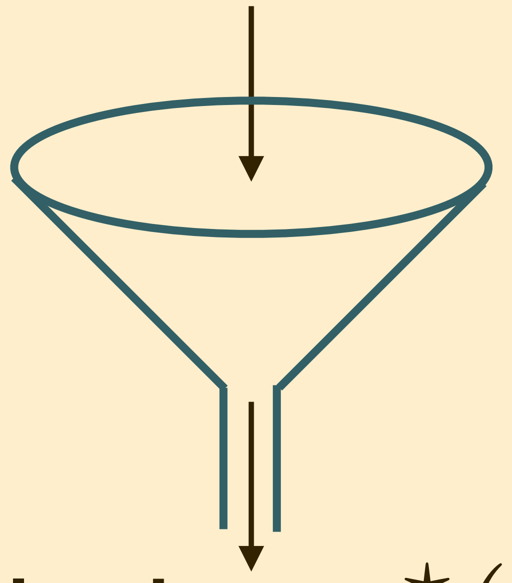
Learned warm start tailored for downstream algorithm

Benefits of our learning framework

End-to-end learning: warm-start predictions tailored to downstream algorithm



Guaranteed convergence

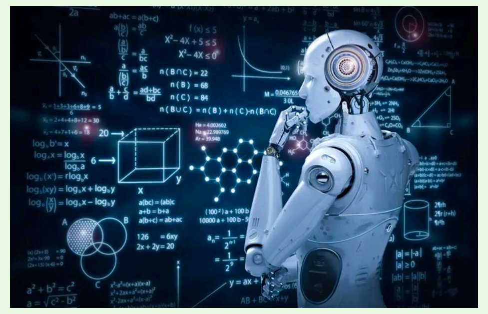


Parameter θ

Learned solver with convergence

Solution $z^\star(\theta)$

Generalization guarantees



I. Guarantees from k training steps to t evaluation steps

II. Guarantees to unseen data

Easy integration with popular solvers

SCS
SPLITTING CONIC SOLVER

Conic programs

$$\begin{array}{ll} \text{minimize} & (1/2)x^T P x + c^T x \\ \text{subject to} & Ax + s = b \\ & s \in \mathcal{K} \end{array}$$

```
sol = scs_solver.solve(warm_start=True,
                       x=x0, y=y0, s=s0)
```

Allows us to quantify solve time in seconds

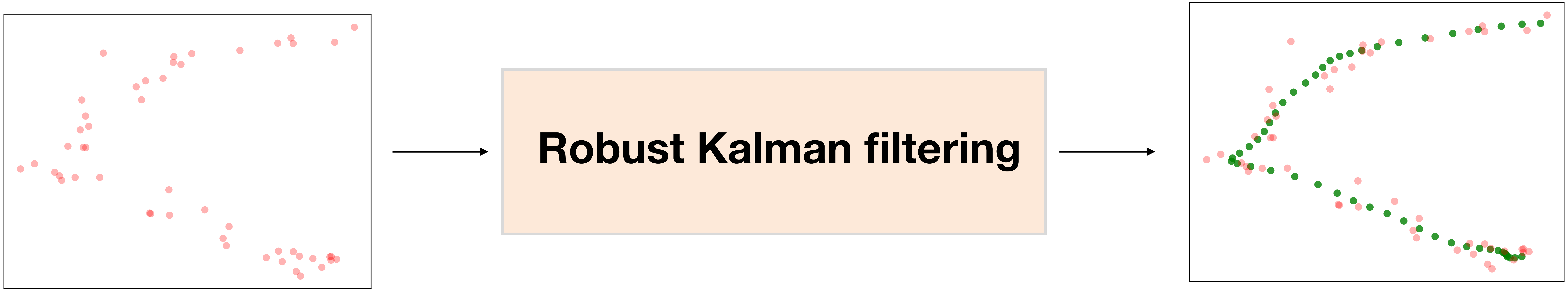
Numerical Experiments

Comparing our learned warm starts 🧠 against

Baseline initializations

1. **Cold-start: initialize at zero** ❄
2. **Nearest neighbor: initialize with solution of nearest training problem**

Robust Kalman filtering



Robust Kalman filtering

$\theta = \{y_t\}_{t=0}^{T-1}$

Noisy trajectory

$\longrightarrow$

Second-order cone program

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{T-1} \|w_t\|_2^2 + \mu\psi_\rho(v_t) \\ \text{subject to} & x_{t+1} = Ax_t + Bw_t \quad \forall t \\ & y_t = Cx_t + v_t \quad \forall t \end{array}$$

$\longrightarrow$

$\{x_t^\star, w_t^\star, v_t^\star\}_{t=0}^{T-1}$

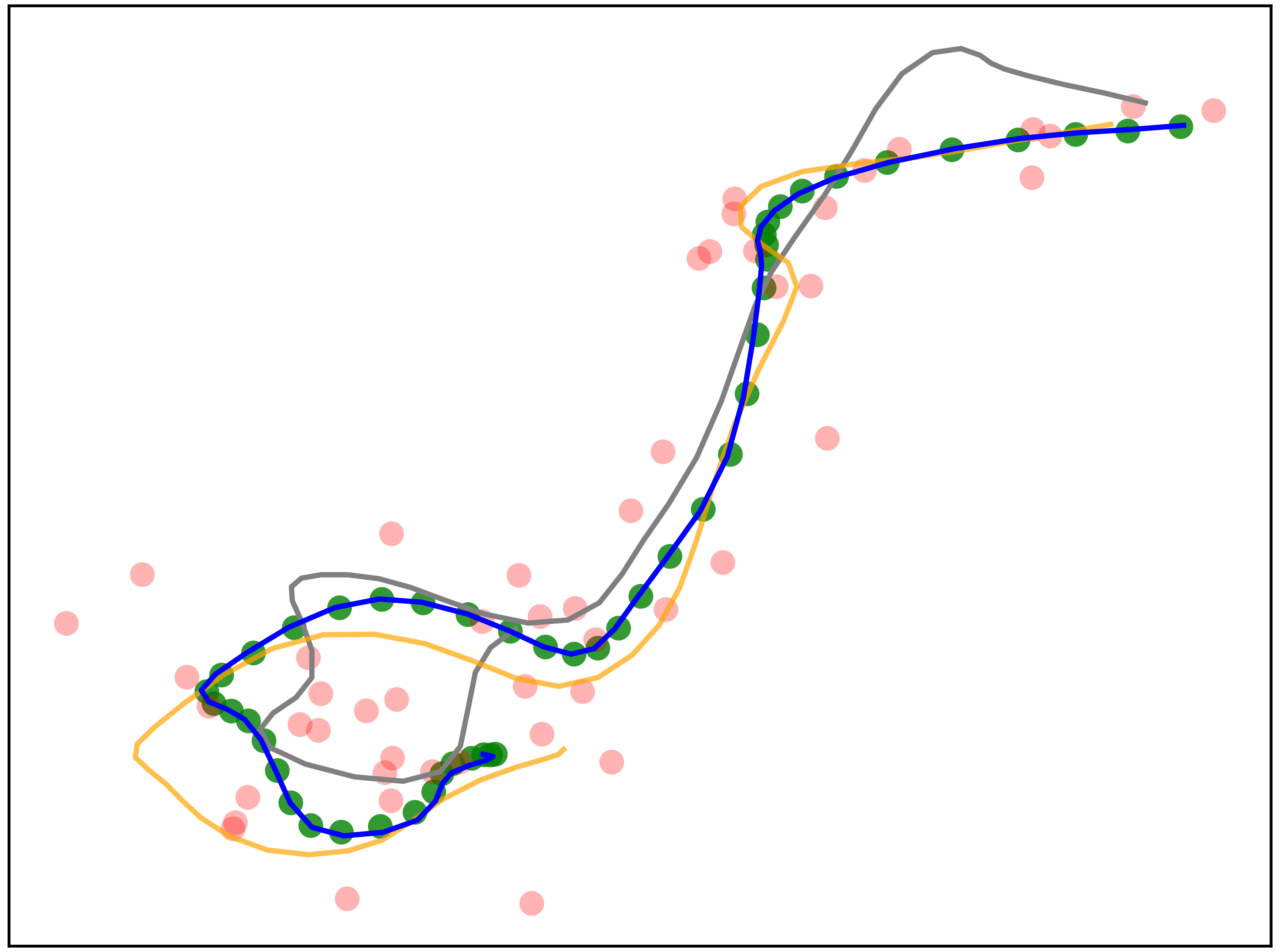
Recovered trajectory

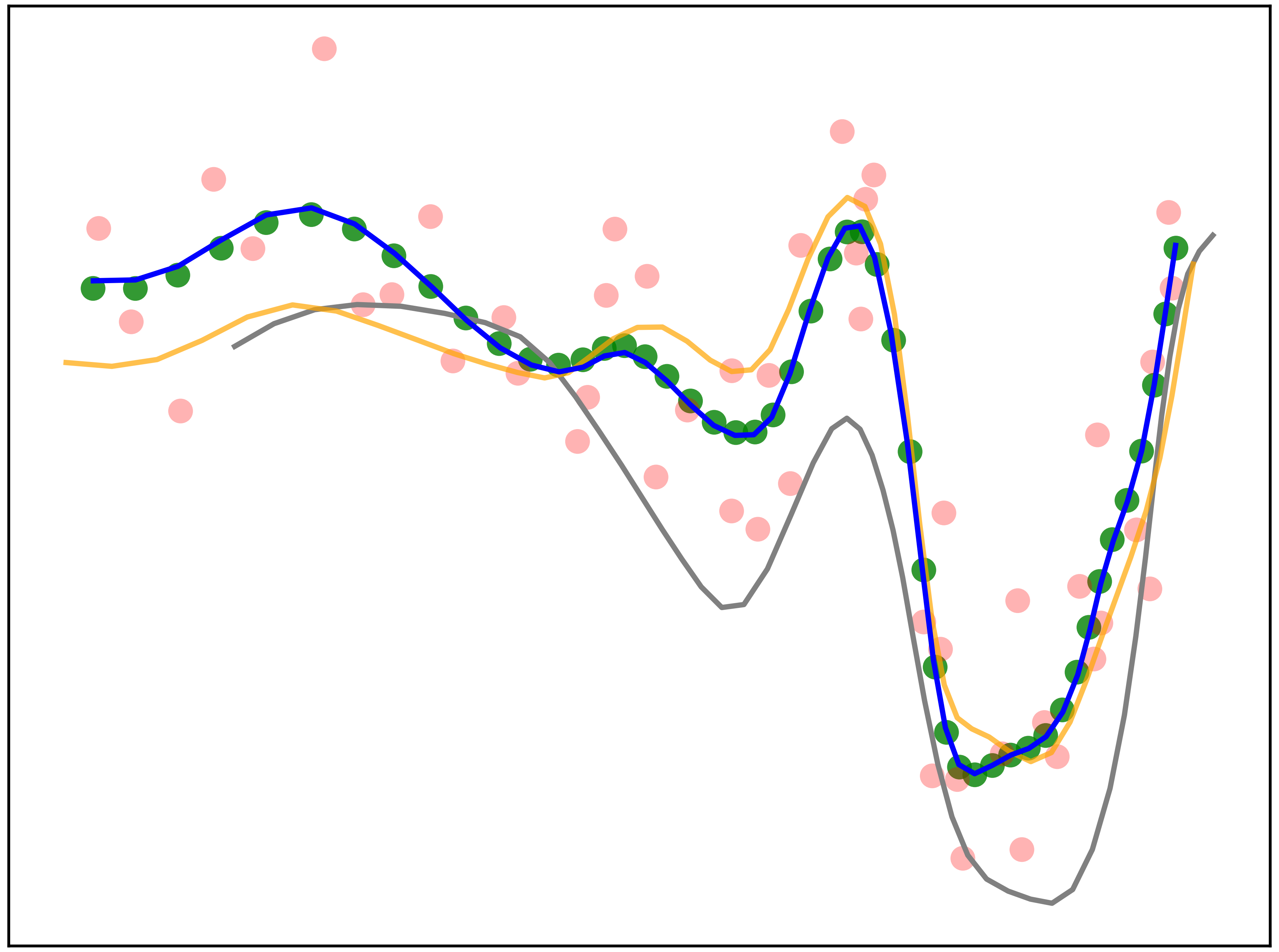
Dynamics matrices: A, B

Observation matrix: C

Huber loss: ψ_ρ

Robust Kalman filtering visuals

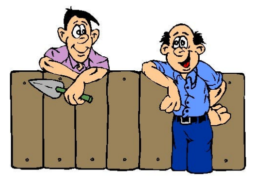




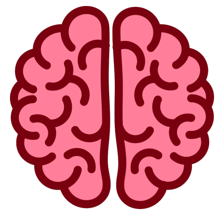
Solution after 5 fixed-point steps
with different initializations

🟥 Noisy trajectory

🟩 Optimal solution

🟨 Nearest neighbor 

⬛ Previous solution 

🟦 Learned: $k = 5$ 

With learning, we can estimate the state well

Model predictive control (MPC) of a quadcopter

Current state,
previous control
reference trajectory

$\longrightarrow$ **Controller** $\longrightarrow$

Control
inputs

$$\theta = (x_{\text{init}}, u_{\text{prev}}, \{x_t^{\text{ref}}\}_{t=1}^T) \longrightarrow$$

Quadratic program

minimize $\sum_{t=1}^T (x_t - x_t^{\text{ref}})^T Q (x_t - x_t^{\text{ref}}) + \sum_{t=0}^{T-1} {u_t}^T R u_t$

subject to

$$x_{t+1} = A(\theta) x_t + B(\theta) u_t$$

$$u_{\min} \le u_t \le u_{\min}$$

$$x_{\min} \le x_t \le x_{\max}$$

$$|u_{t+1} - u_t| \le \Delta u$$

$$x_0 = x_{\text{init}}$$

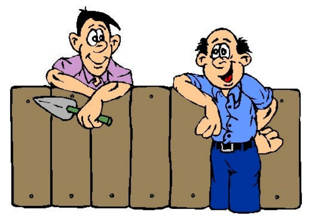
$$u_{-1} = u_{\text{prev}}$$

$$\longrightarrow \{x_t^\star, u_t^\star\}_{t=0}^T$$

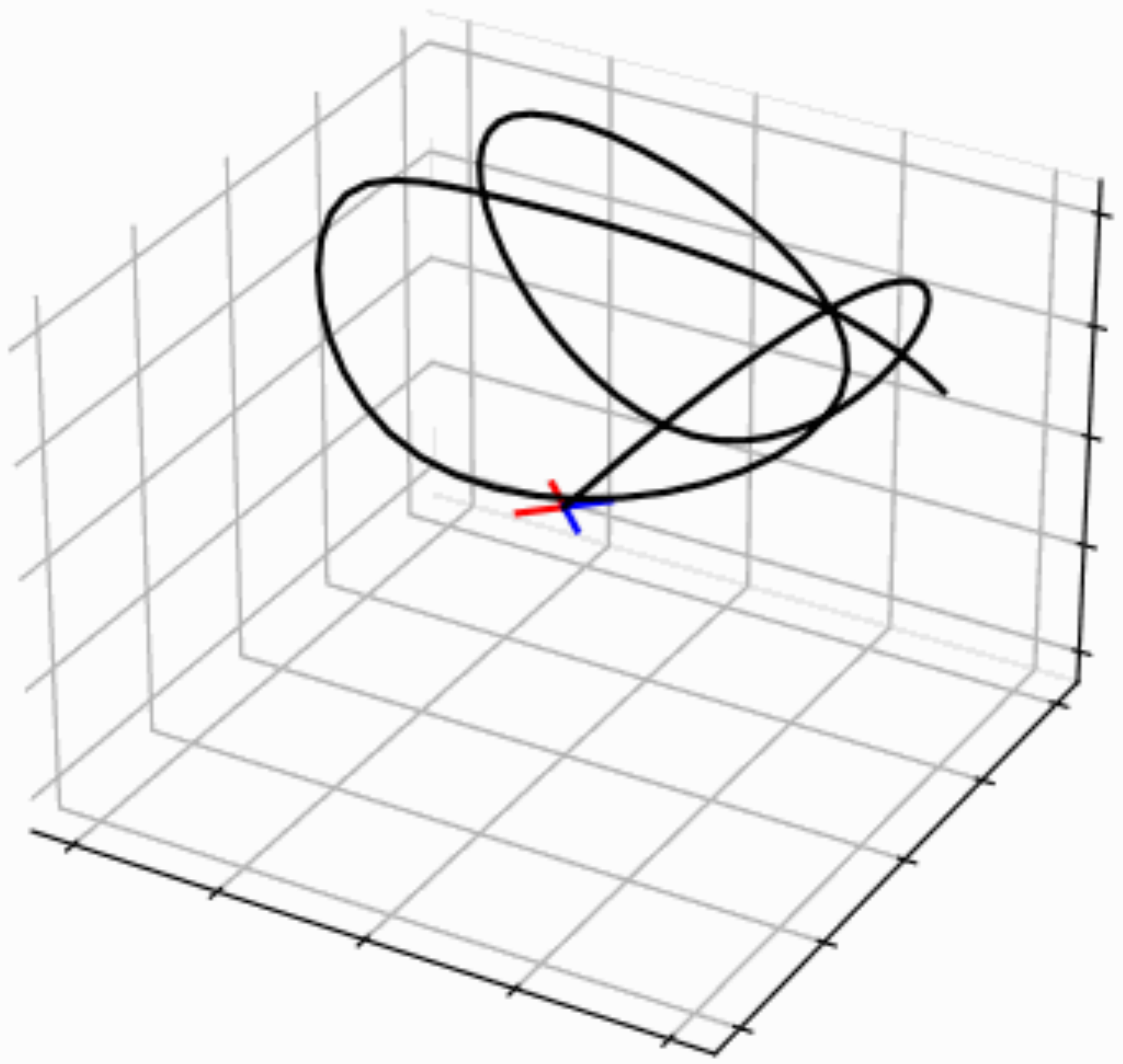
Linearized dynamics

MPC of a quadcopter in a closed loop

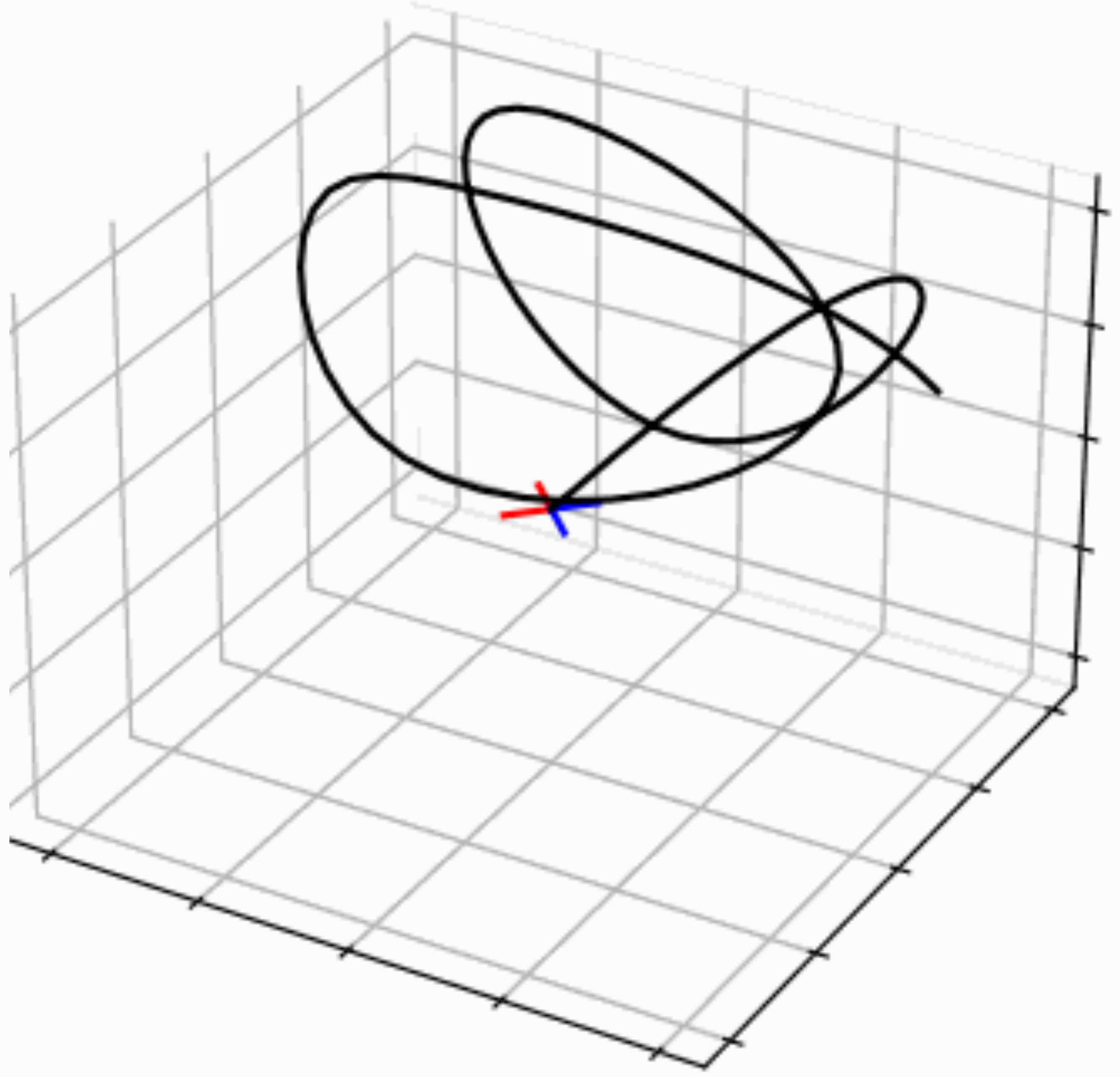
Budget of 15 fixed-point steps



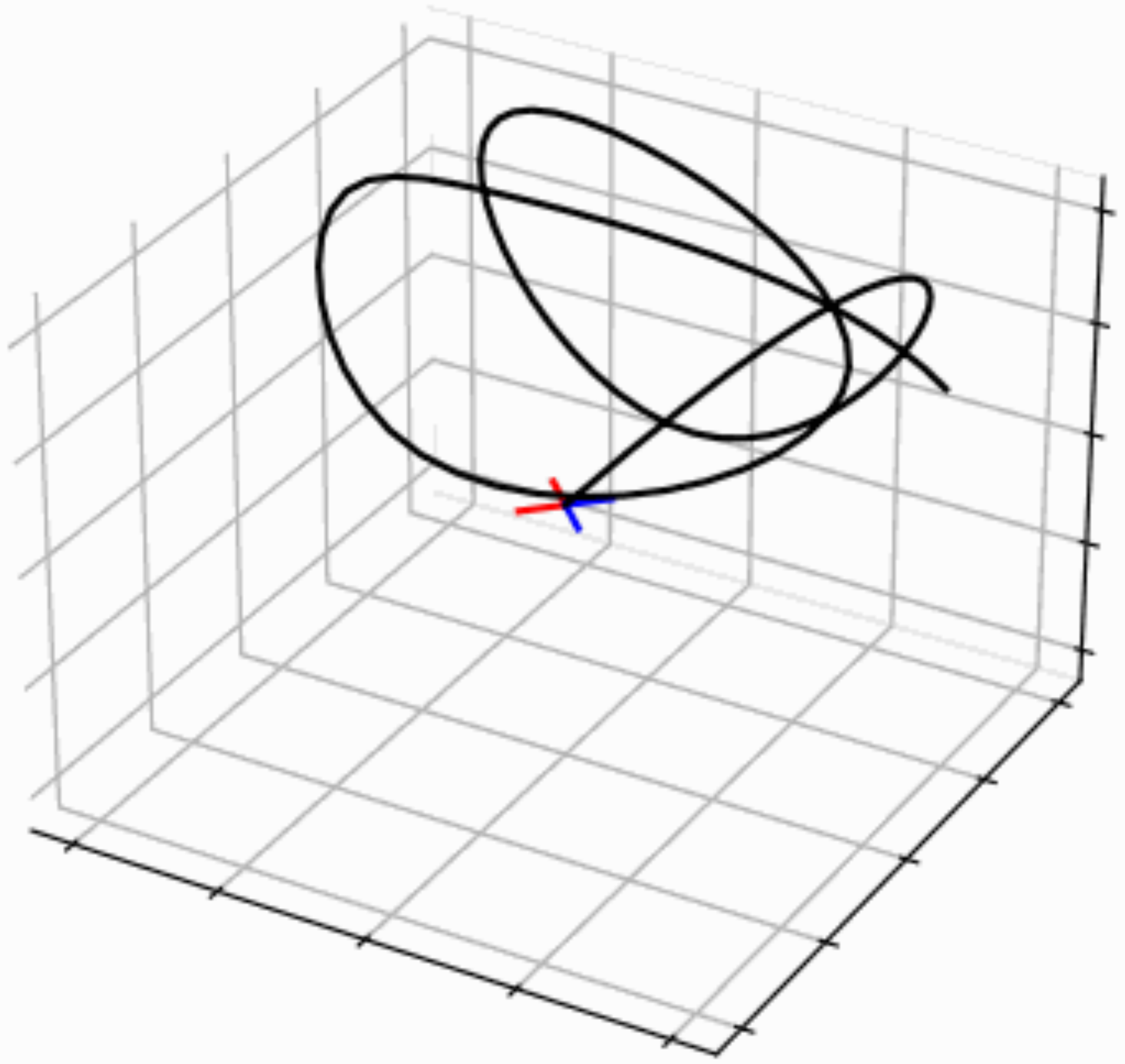
Nearest neighbor



Previous solution



Learned: $k = 5$



With learning, we can track the trajectory well

Image deblurring


Image deblurring

$\theta = b$

Blurred image

Quadratic program

minimize $\quad \|Ax - b\|_2^2 + \lambda \|x\|_1$

subject to $\quad 0 \leq x \leq 1$

$x^\star$

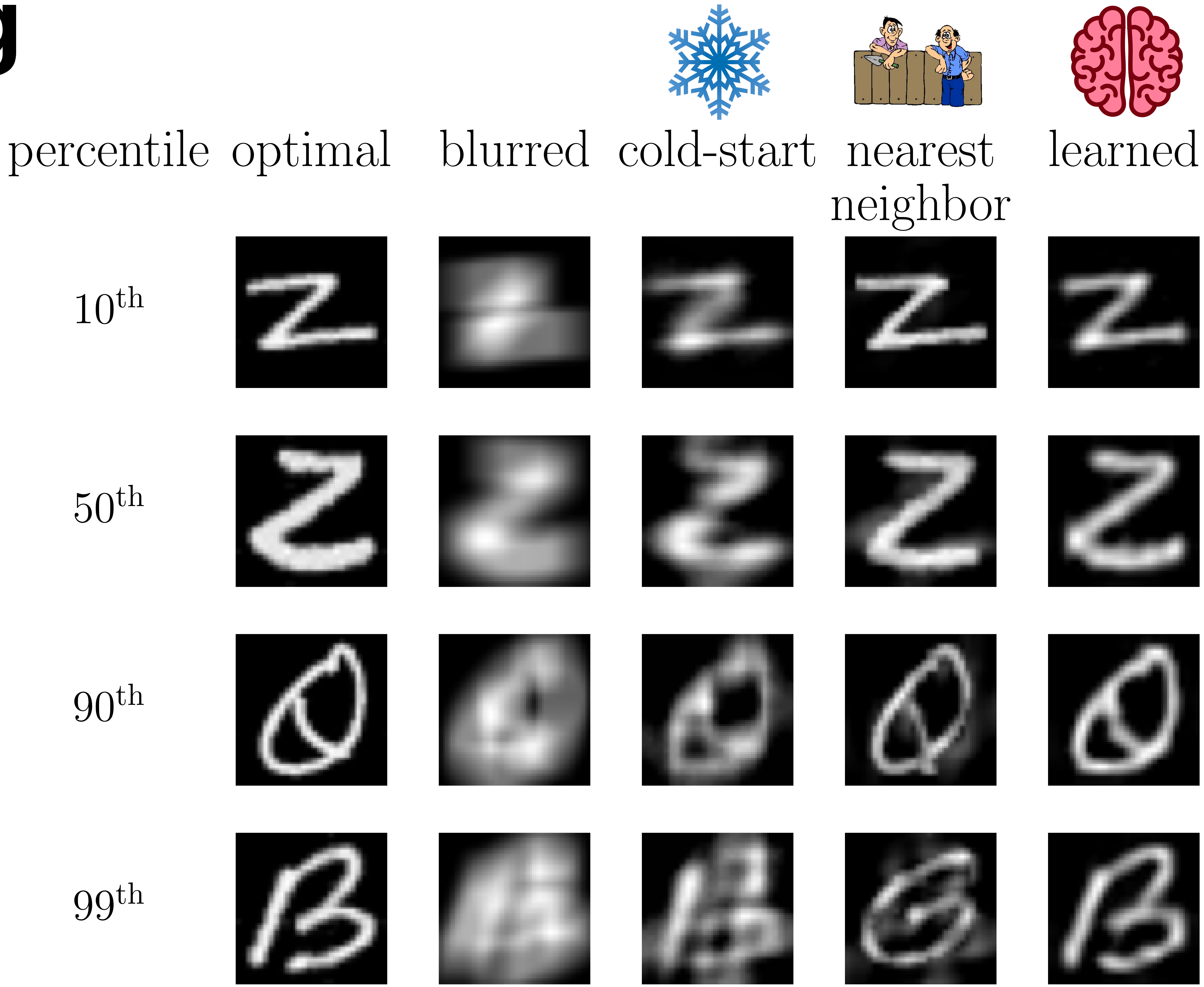
Deblurred image

A: blur operator

Image deblurring

50 fixed-point steps

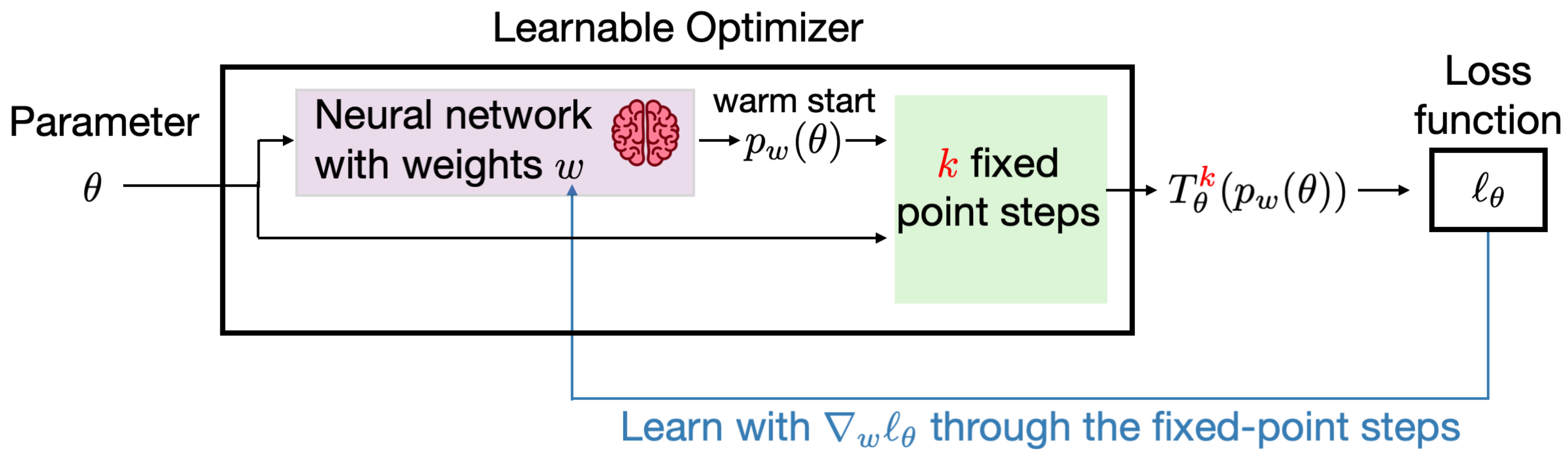
Distance to nearest neighbor increases



With learning, we can deblur all of the images quickly

Talk Outline

- **Part 1: Learning to Warm-Start Fixed-Point Optimization Algorithms**
- **Part 2: Practical Performance Guarantees for Classical and Learned Optimizers**



Learnable Optimizer

Parameter

θ

Neural network with weights w

warm start

$p_w(\theta)$

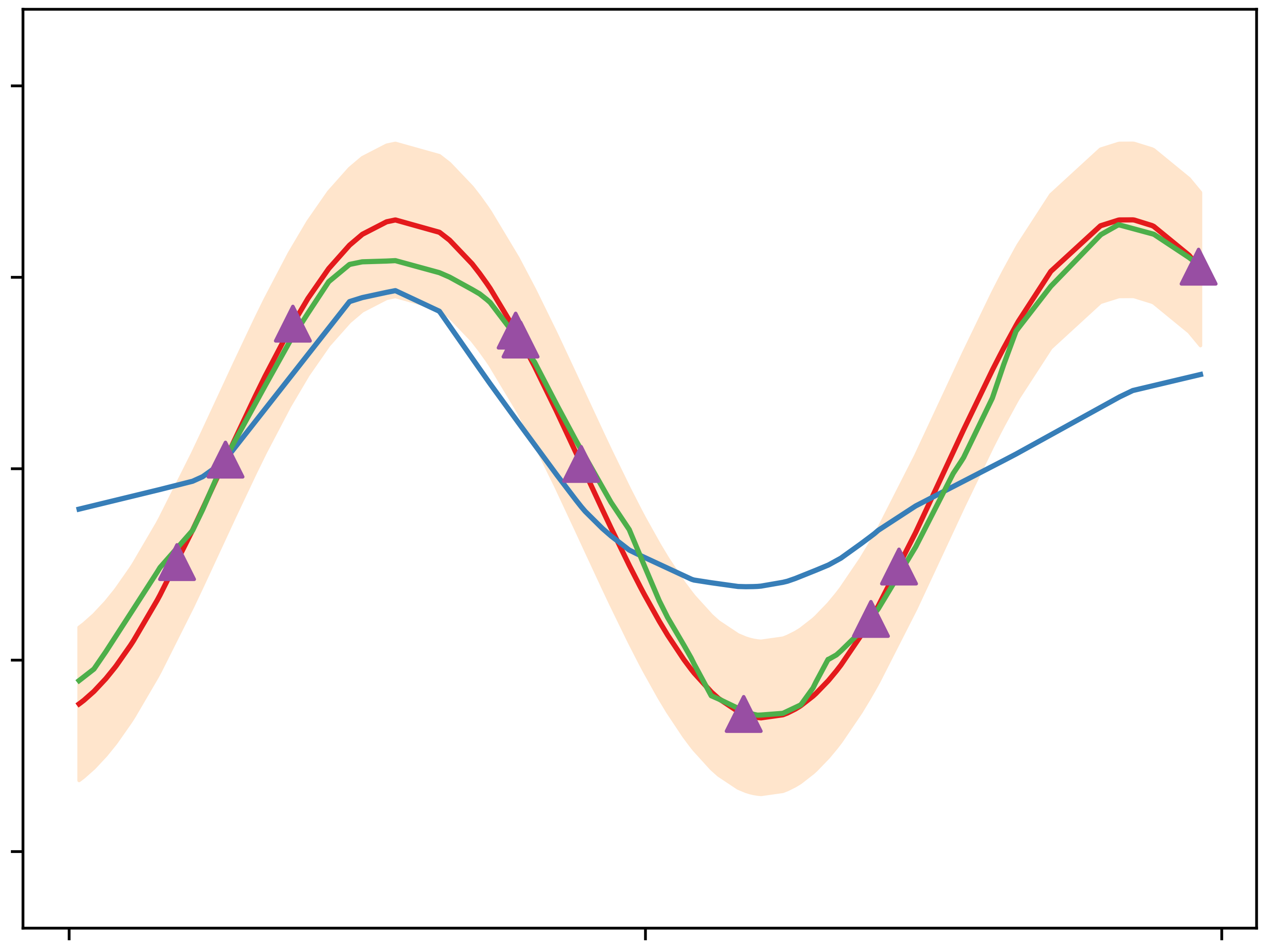
k fixed point steps

$T_\theta^k(p_w(\theta))$

Loss function

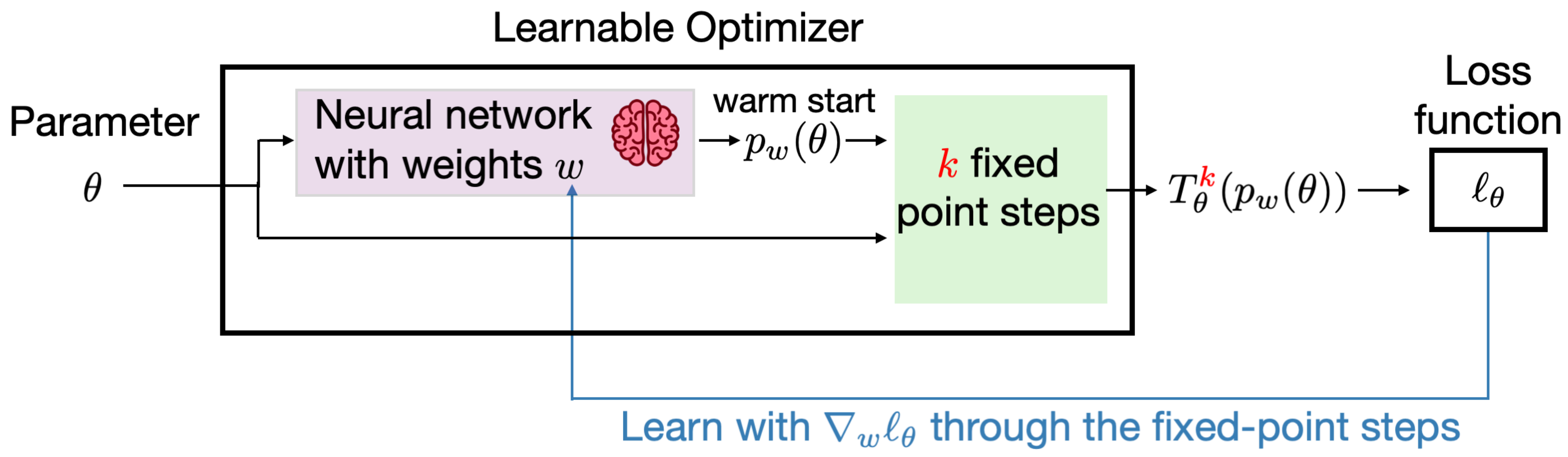
ℓ_θ

Learn with $\nabla_w \ell_\theta$ through the fixed-point steps



Talk Outline

- **Part 1: Learning to Warm-Start Fixed-Point Optimization Algorithms**



Learnable Optimizer

Parameter

θ

Neural network with weights w

warm start

$p_w(\theta)$

k fixed point steps

$T_\theta^k(p_w(\theta))$

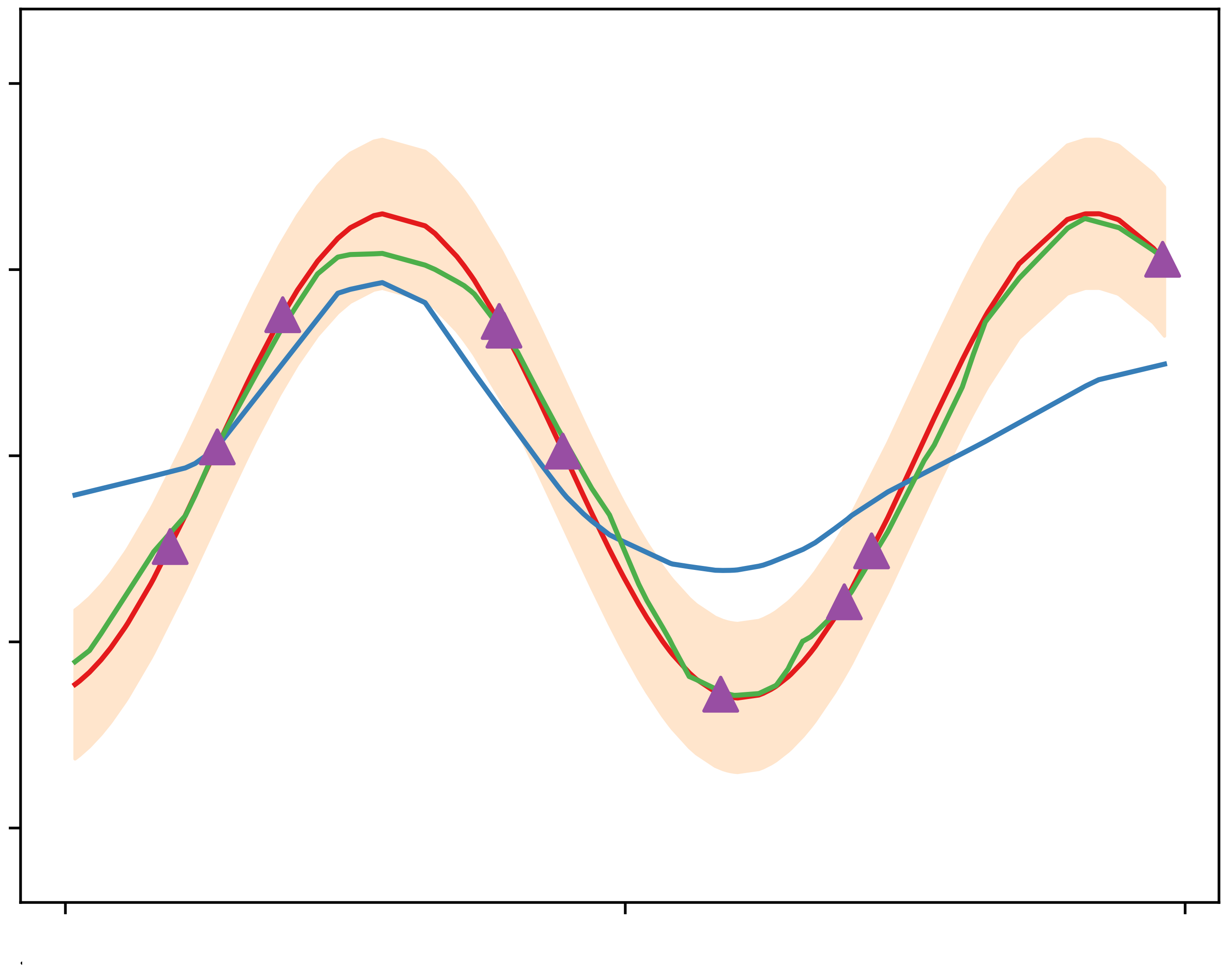
Loss function

ℓ_θ

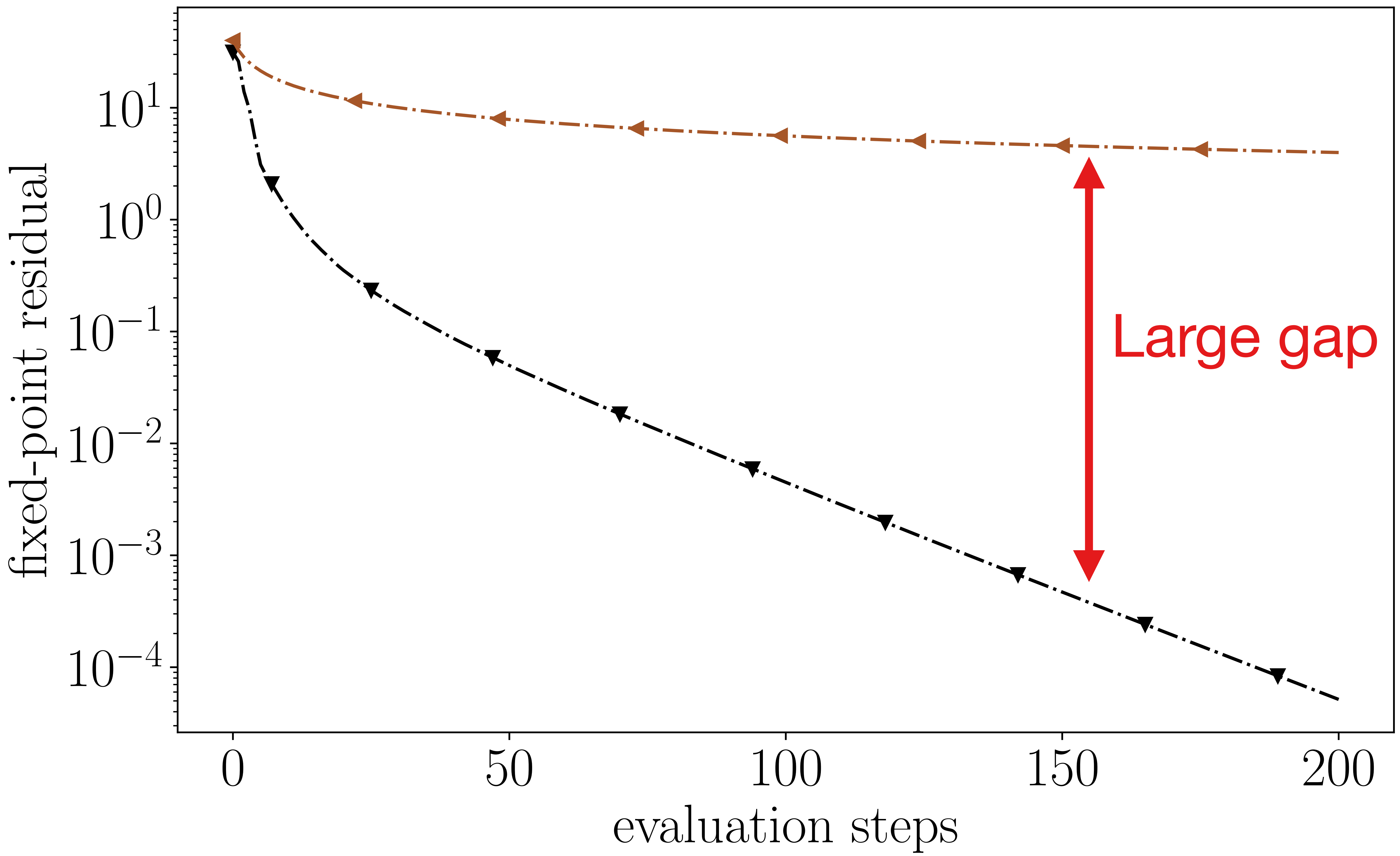
Learn with $\nabla_w \ell_\theta$ through the fixed-point steps

- **Part 2: Practical Performance Guarantees for Classical and Learned Optimizers**

Classical = no learning



Worst-case bounds can be very loose



Example: robust Kalman filtering

Second-order cone program

$$\begin{array}{ll} \text{minimize} & \sum_{t=0}^{T-1} \|w_t\|_2^2 + \mu\psi_\rho(v_t) \\ \text{subject to} & x_{t+1} = Ax_t + Bw_t \quad \forall t \\ & y_t = Cx_t + v_t \quad \forall t \end{array}$$

▼ SCS empirical average performance over 1000 parametric problems

◀ Worst-case bound

In practice: **linear** convergence over the parametric family

Worst-case analysis: **sublinear** convergence

Worst-case bounds do not consider the **parametric** structure

Approach: solve N problems and then bound

We will bound 0-1 error metrics

We will provide guarantees for any measured quantity

algorithm steps

tolerance

$$e(\theta) = \mathbf{1}(\ell^{k}(\theta) > \epsilon)$$

Standard metrics

e.g., fixed-point residual

algorithm steps

cold start

tolerance

$$e(\theta) = \mathbf{1}(\ell_\theta^{\text{fp}}(T_\theta^{k}(0)) > \epsilon)$$

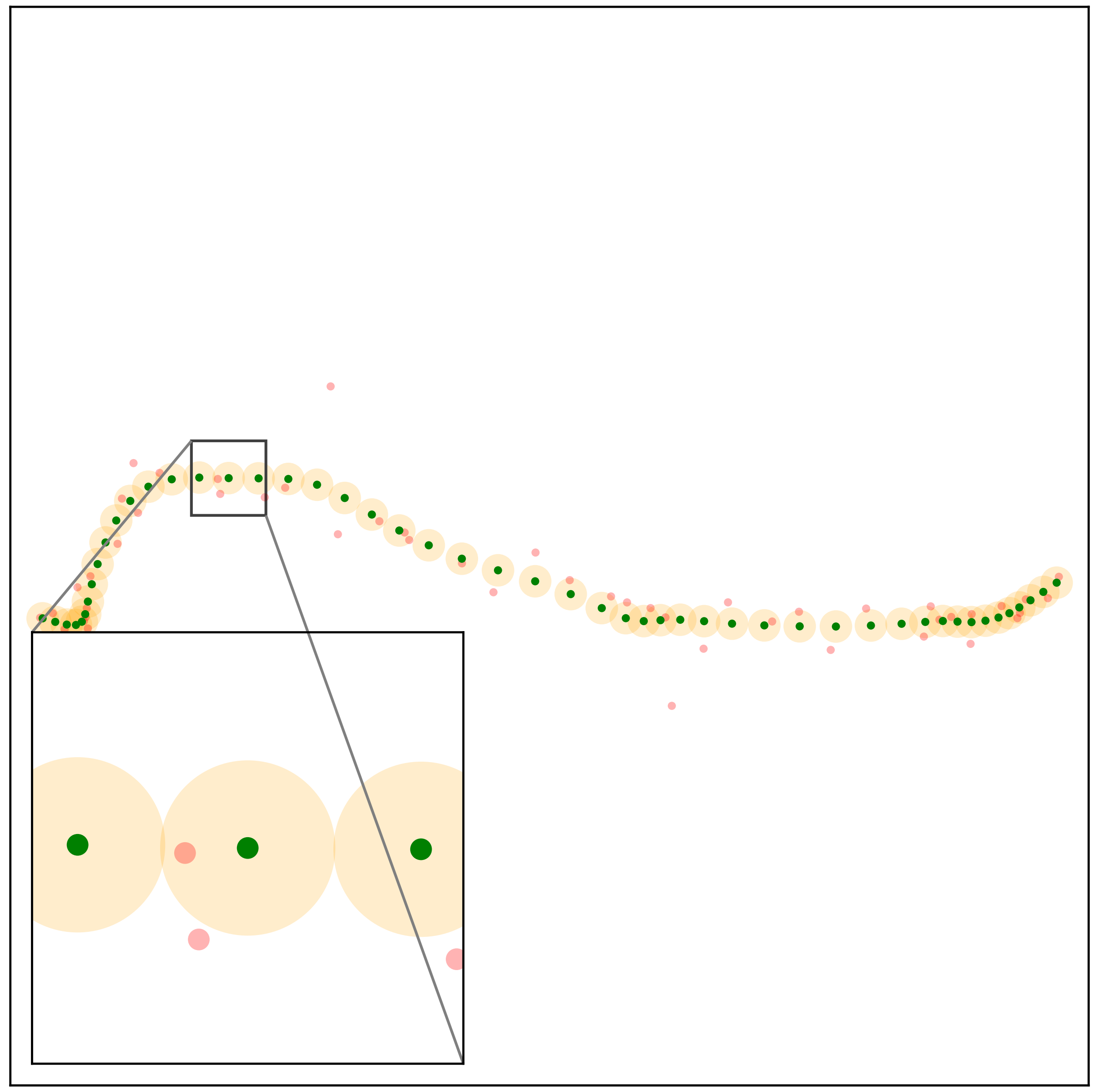
Task-specific metrics:

e.g., quality of extracted states in robust Kalman filtering

recovered state

optimal state

$$e(\theta) = \mathbf{1}\left(\max_{t=1,\ldots,T} \|x_t - x_t^\star\|_2 > \epsilon\right)$$



Background: Kullback-Liebler Divergence

KL divergence: measures distance between distributions

$$\mathrm{KL}(q \parallel p) = \sum_{i=1}^{m} q_i \log\left(\frac{q_i}{p_i}\right)$$

Our bounds on the risk will take the form

$\mathrm{KL}($empirical risk $\parallel$ risk$) \leq$ regularizer

Invert these bounds by solving

risk $\leq \mathrm{KL}^{-1}$ (empirical risk $\mid$ regularizer)

$$\mathrm{KL}^{-1}(q \mid c) = \begin{array}{ll} \text{maximize} & p \\ \text{subject to} & q \log \frac{q}{p} + (1-q) \log \frac{1-q}{1-p} \leq c \\ & 0 \leq p \leq 1 \end{array}$$

1D convex optimization problem

Statistical learning theory can provide probabilistic guarantees

algorithm steps

tolerance

$$e(\theta) = \mathbf{1}(\ell^k(\theta) > \epsilon)$$

Sample convergence bound: with probability $1 - \delta$ [Langford et. al 2001]

$$\mathbf{E}_{\theta \sim \mathcal{X}} e(\theta) \leq \mathrm{KL}^{-1}\left(\frac{1}{N}\sum_{i=1}^{N} e(\theta_i) \,\middle|\, \frac{\log(2/\delta)}{N}\right)$$

Number of problems

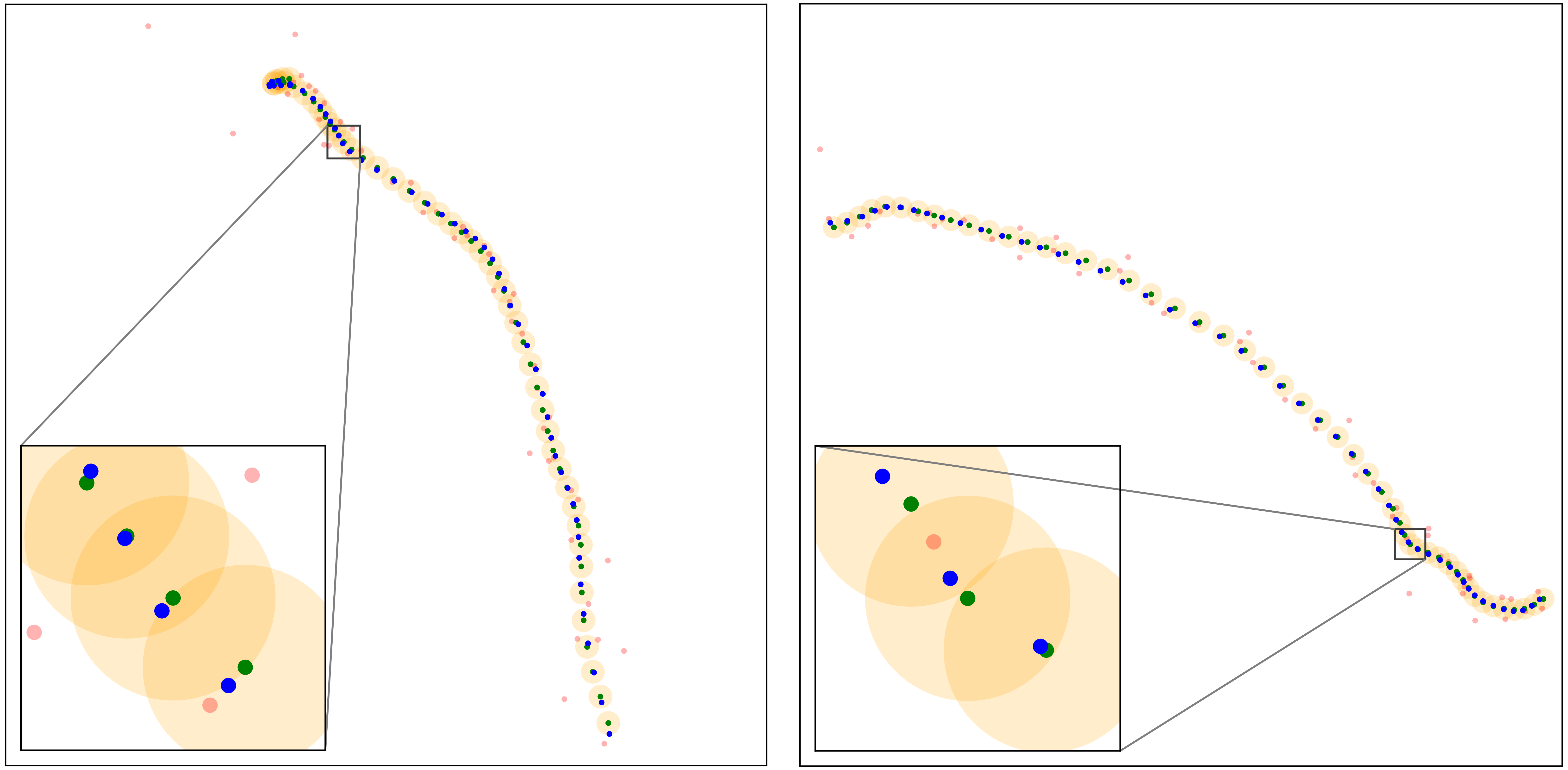
$\mathbf{P}(\ell^k(\theta) > \epsilon) =$ risk $\leq \mathrm{KL}^{-1}$ (empirical risk | regularizer)

”With probability $1-\delta$, 90% of the time the fixed-point residual is below $\epsilon = 0.01$ after $k = 20$ steps”

Robust Kalman filtering guarantees

With 1000 samples, we provide strong probabilistic guarantees on the 99th quantile

Visualizing Robust Kalman filtering guarantees



Task-specific error metric

$$e(\theta) = \mathbf{1}\left(\max_{t=1,\ldots,T} \|x_t - x_t^\star\|_2 > \epsilon\right)$$

🟥 Noisy trajectory

🟩 Optimal solution

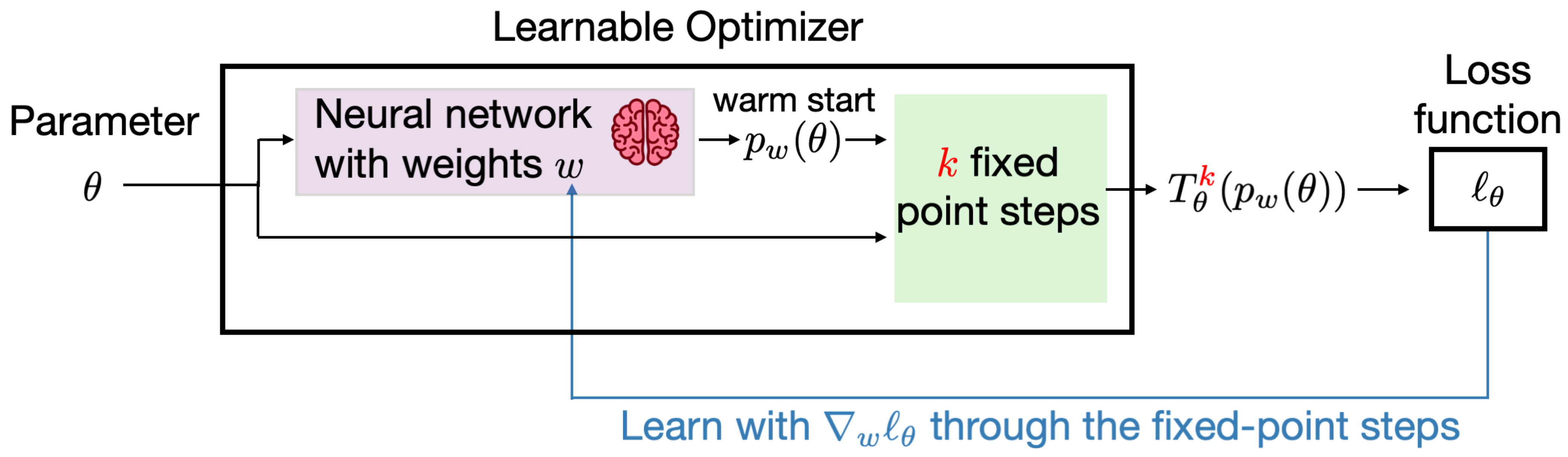
🟦 Solution after 15 steps

🟨 Region with guarantee

"With high probability, 90% of the time, all of the recovered states after 15 steps of problems drawn from the distribution will be within the correct ball with radius 0.1"

Talk Outline

- **Part 1: Learning to Warm-Start Fixed-Point Optimization Algorithms**
- **Part 2: Practical Performance Guarantees for Classical and Learned Optimizers**


Learnable Optimizer

Parameter

θ

Neural network with weights w

warm start

$p_w(\theta)$

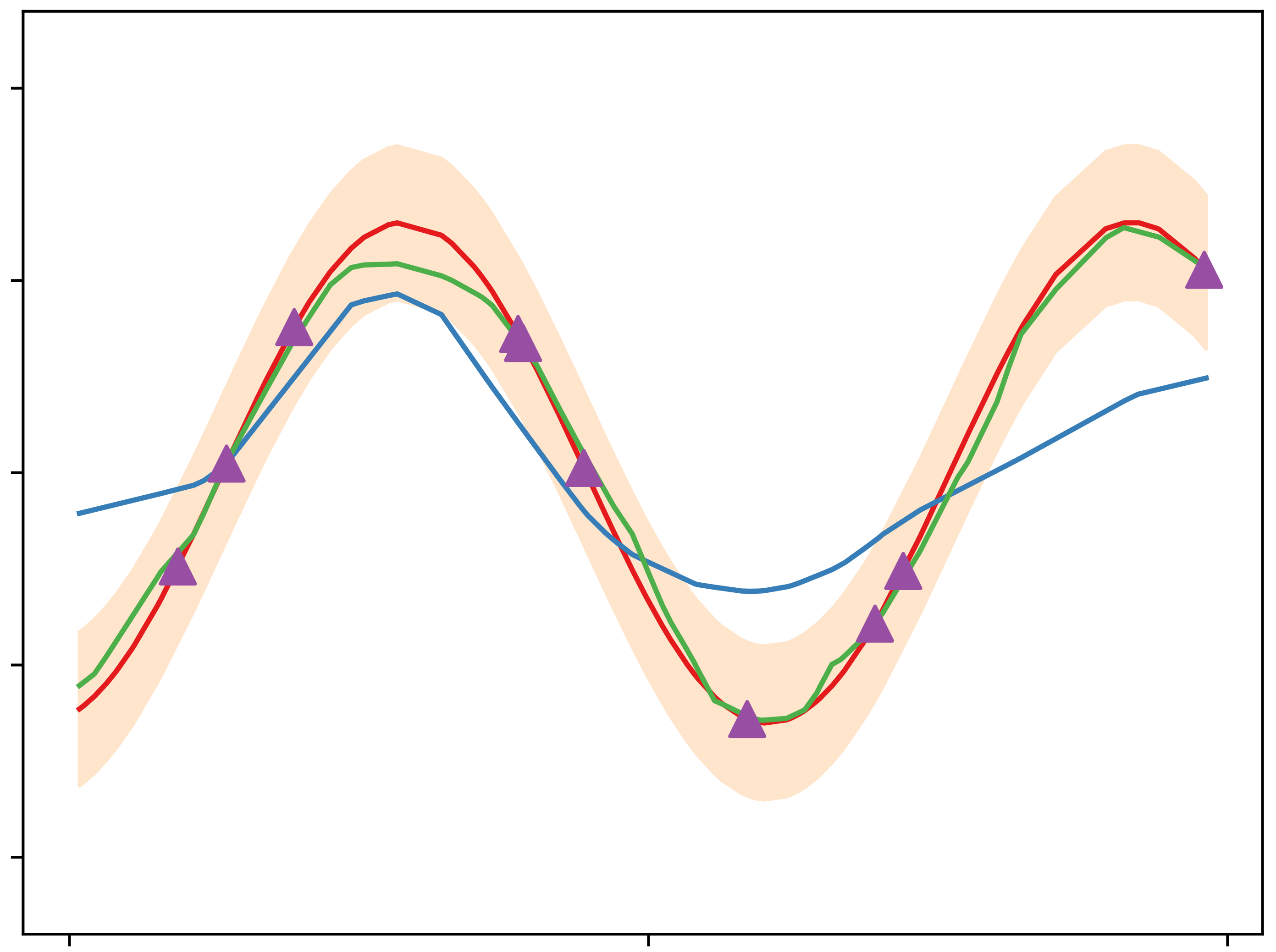
k fixed point steps

$T_\theta^k(p_w(\theta))$

Loss function

ℓ_θ

Learn with $\nabla_w \ell_\theta$ through the fixed-point steps



Tutorial on Amortized Optimization [Amos 2023]

"Despite having the capacity of surpassing the convergence rates of other algorithms, oftentimes in practice amortized optimization methods can deeply struggle to generalize and converge to reasonable solutions."

PAC-Bayes guarantees for learned optimizers

algorithm steps

tolerance

$$e_w(\theta) = \mathbf{1}(\ell_w^k(\theta) > \epsilon)$$

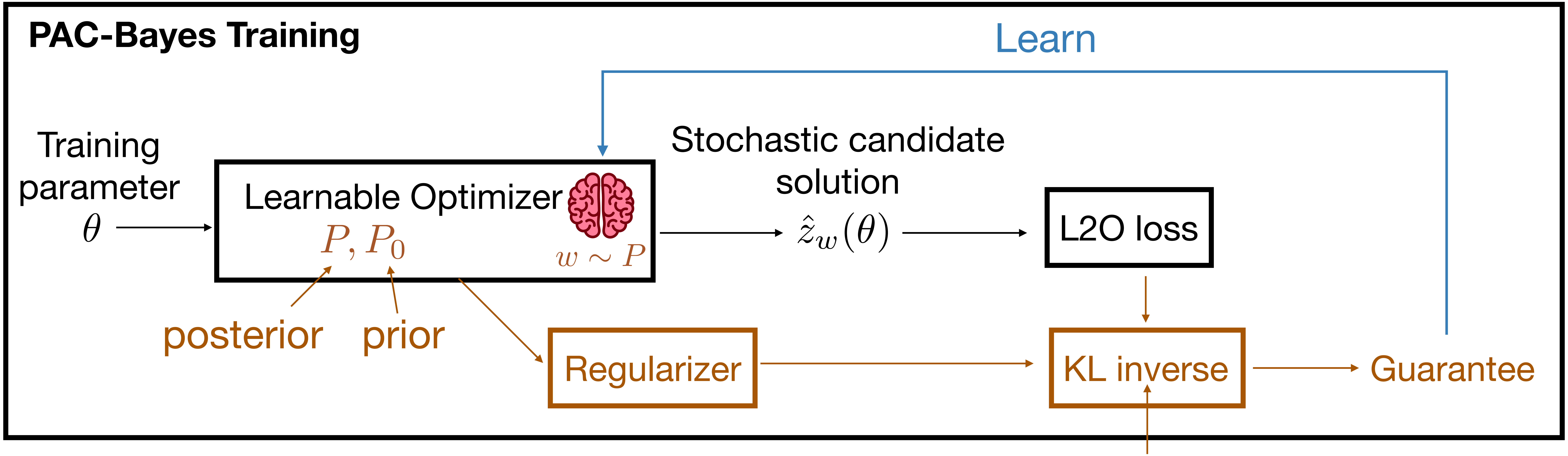
learnable weights

McAllester bound: given posterior and prior distributions [McAllester et. al 2003]
P and P_0, with probability $1-\delta$

$$\begin{array}{r} \mathbf{E}_{\theta\sim\mathcal{X}}\mathbf{E}_{w\sim P}\, e_w(\theta) \le \\[6ex] \text{risk} \le \end{array} \boxed{\begin{array}{l} \mathrm{KL}^{-1}\left(\frac{1}{N}\sum_{i=1}^{N} \mathbf{E}_{w\sim P}\, e_w(\theta_i) \middle| \frac{1}{N}\left(\mathrm{KL}(\mathrm{P} \parallel \mathrm{P}_0) + \log(\mathrm{N}/\delta)\right)\right) \\[6ex] \mathrm{KL}^{-1}\left(\text{empirical risk} \mid \text{regularizer}\right) \end{array}}$$

Optimize the bounds directly

PAC-Bayes training architecture to optimize the guarantees



Use differentiable optimization

We show that the derivative always exists

We implement the learnable optimizer and train with this architecture

Learned algorithms for sparse coding

Noisy measurements
$\theta = b$

→

Sparse coding

Recover sparse $z^\star$ from $b = Dz^\star + \sigma$

→

Ground truth sparse signal
$z^\star$

D: dictionary, σ: noise

Standard technique

minimize $\|Dz - b\|_2^2 + \lambda \|z\|_1$

ISTA (iterative shrinkage thresholding algorithm)

(Classical optimizer)

$$z^{j+1} = \text{soft threshold}_{\frac{\lambda}{L}}\left(z^j - \frac{1}{L}D^T(Dz^j - b)\right)$$

Learned ISTA

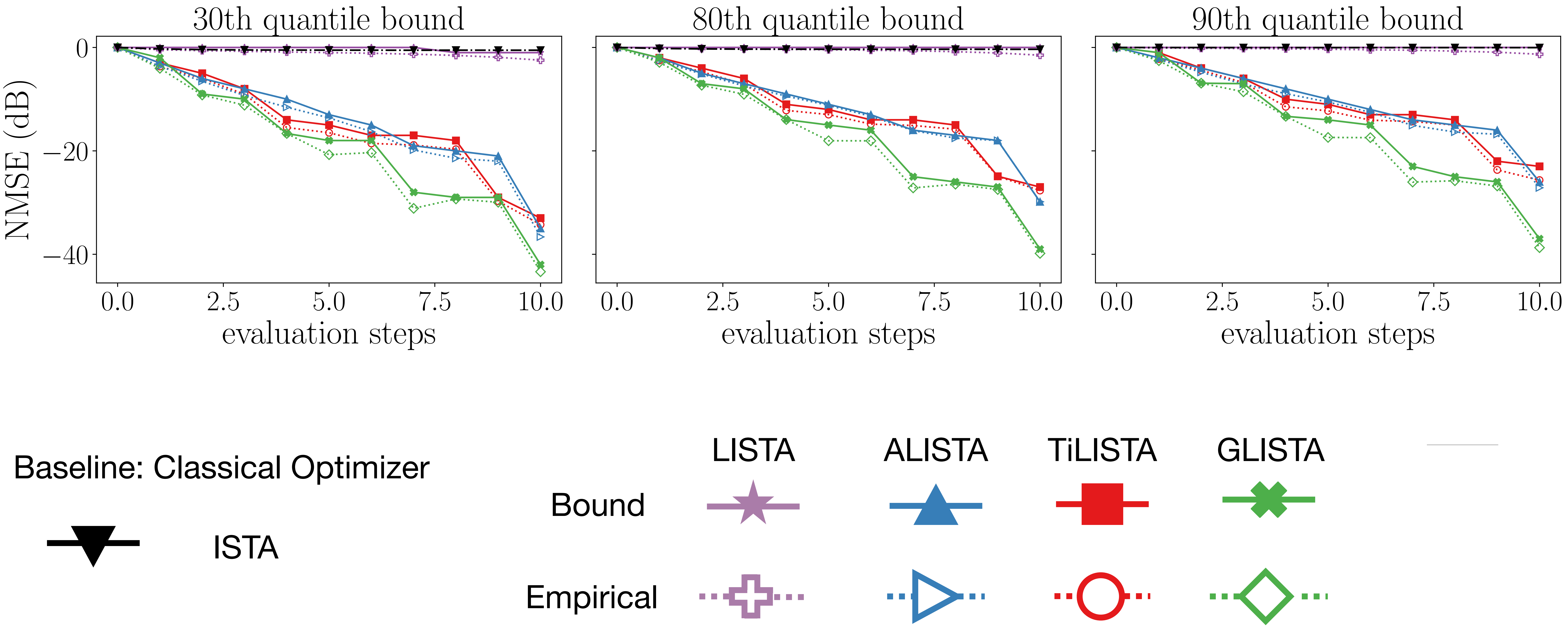
(Learned optimizer)

$$z^{j+1} = \text{soft threshold}_{\psi^j}\left(W_1^j z^j + W_2^j b\right)$$

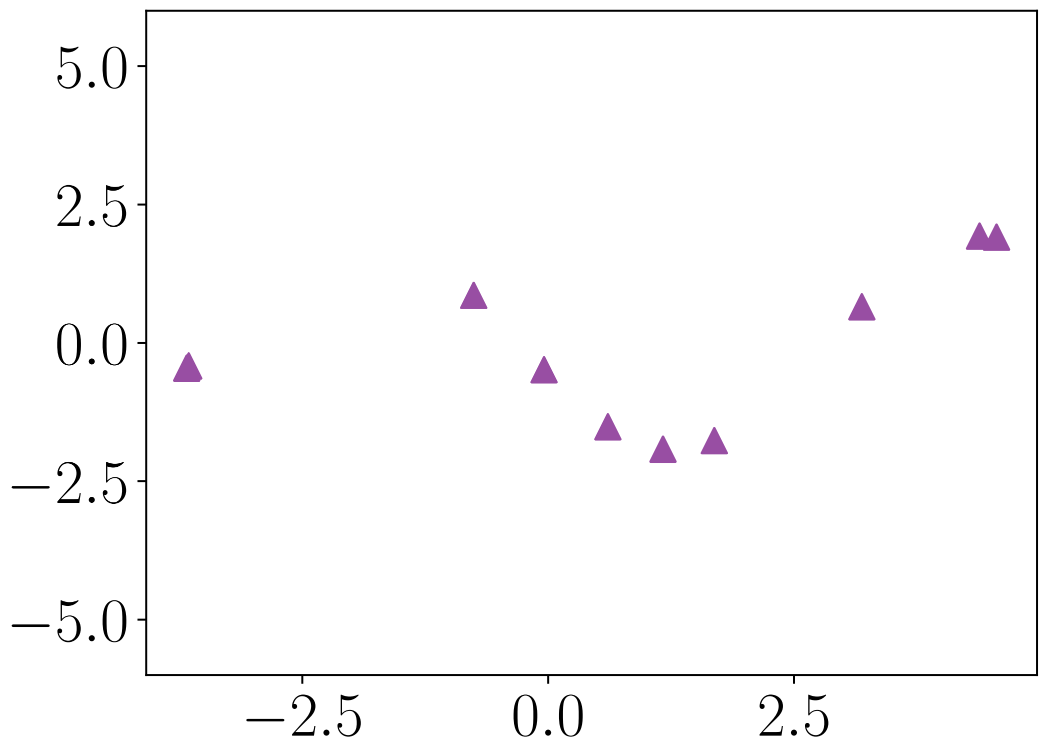
+ variants [Gregor and LeCun 2010, Liu et. al 2019]

soft threshold$_\psi(z) = $ **sign**$(z)\max(0, |z| - \psi)$

Learned ISTA results for sparse coding



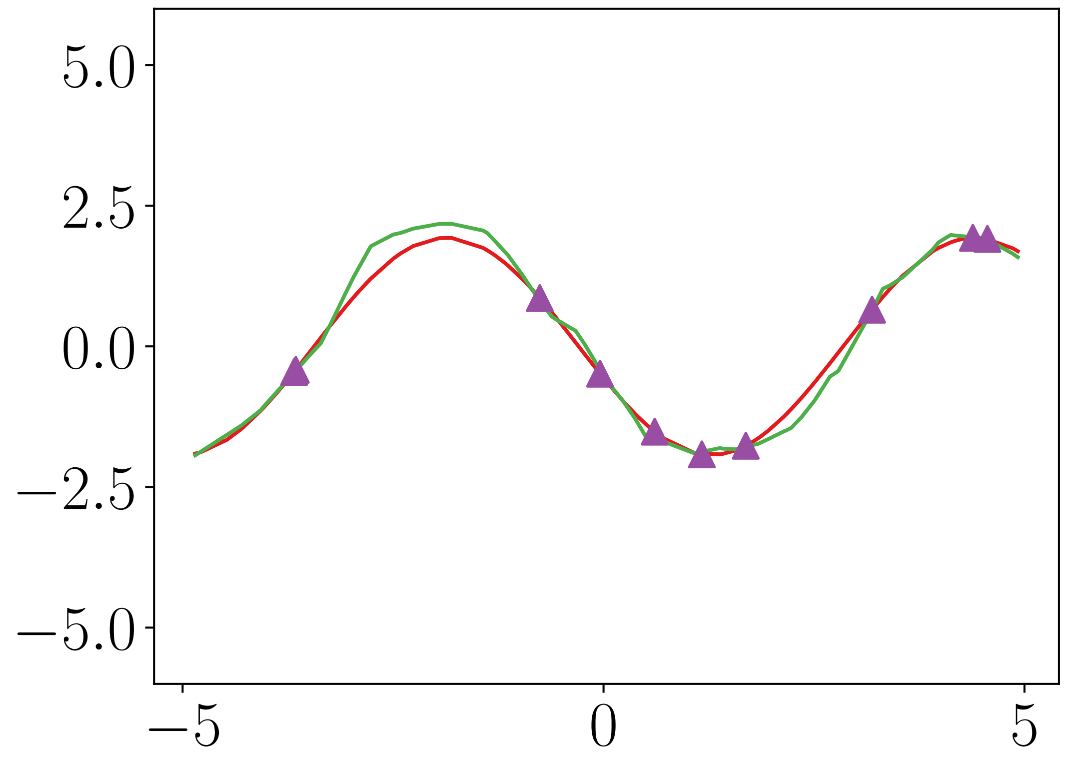
K-shot Meta-Learning for Sine Curves



Neural network learning

find weights z so that $g_z(x_i) \approx y_i$

predictor with weights z



Training dataset with K points

$\mathcal{D}^{\text{train}}$

Gradient step

$$\hat{z} = z - \alpha \nabla_z \mathcal{L}(z, \mathcal{D}^{\text{train}})$$

Weights that generalize to new points quickly

$\hat{z}$

Model-Agnostic Meta-Learning (MAML) [Finn et. al 2017]

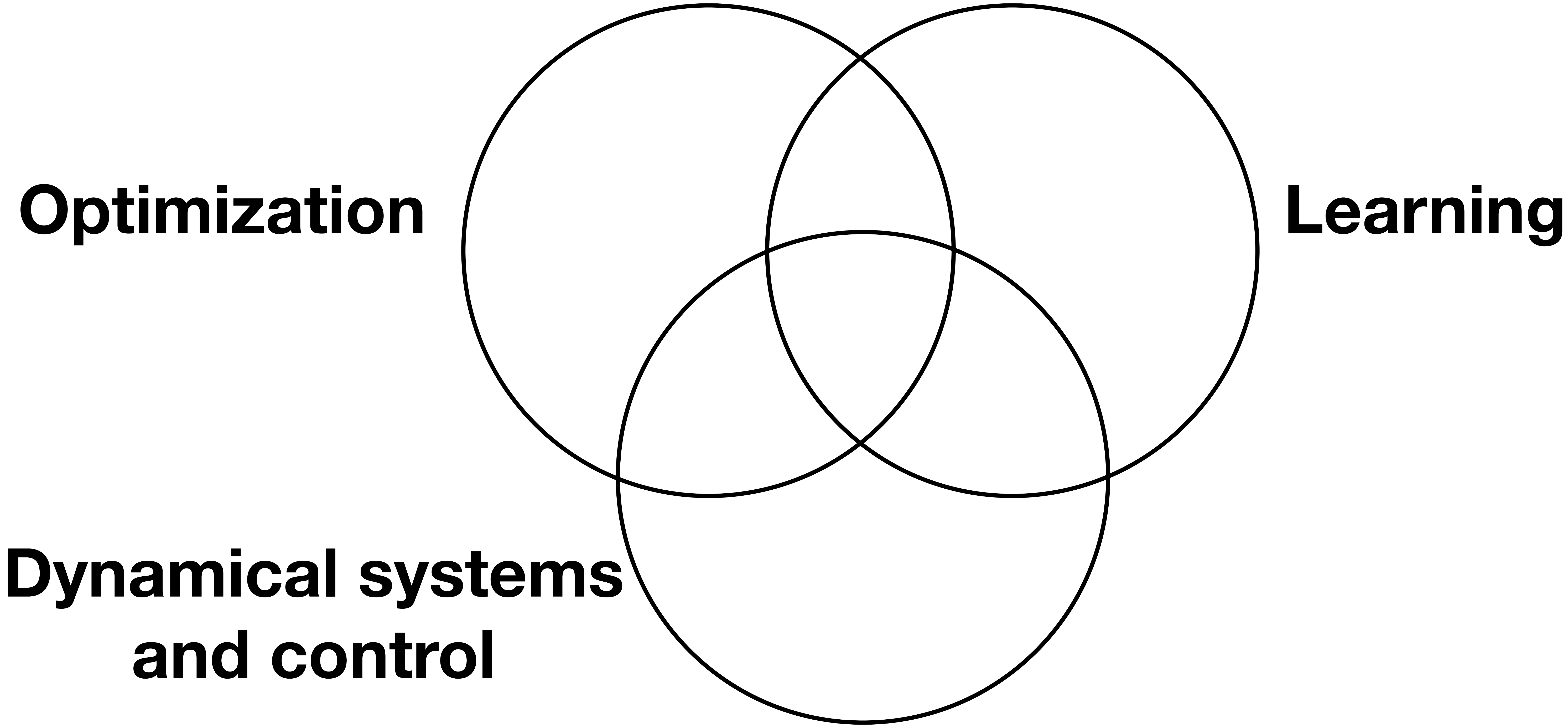
MAML learns a shared initialization z so that $\hat{z}$ performs well on test data

Visualizing Guarantees: K-shot Meta-Learning for Sine Curves

With high probability, 90% of the time stochastic MAML after 10 steps will stay within the band

The pretrained baseline only stays within the band 30% of the time

Future directions



Connections with Computational Robotics Lab

Learning dynamical systems, certificates for stability and safety

Learning to optimize for robotics

Focus on guarantees

Conclusions

We do not need to sacrifice **guarantees for learning-based systems**

Learning to Warm-Start
Fixed-Point Optimization Algorithms

**Journal of Machine Learning Research
(accepted conditioned
on minor revision)**
https://arxiv.org/pdf/2309.07835.pdf

End-to-End Learning to Warm-Start for
Real-Time Quadratic Optimization

**Learning for Dynamics and
Control Conference**
https://arxiv.org/pdf/2212.08260.pdf

Practical Performance Guarantees
for Classical and Learned Optimizers

To be on Arxiv soon!

✉ rajivs@princeton.edu

🌐 rajivsambharya.github.io