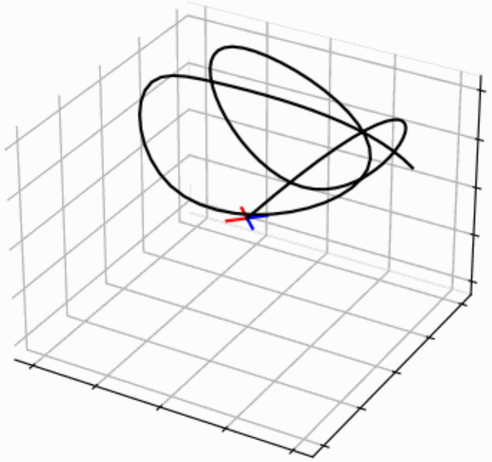
# Learning to Accelerate Optimizers with Guarantees

Penn Talk 2024

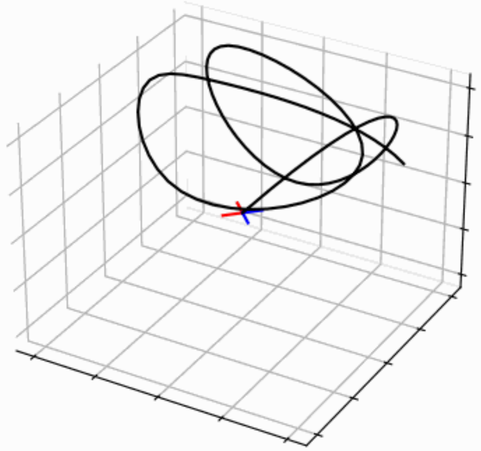
Rajiv Sambharya


# 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**

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{t=1}^{T} \|x_t - x_t^{\text{ref}}\|_2^2 \\ \text{subject to} \quad & x_{t+1} = Ax_t + Bu_t \\ & x_t \in \mathcal{X}, \quad u_t \in \mathcal{U} \\ & x_0 = x_{\text{init}} \end{aligned}$$

Control inputs

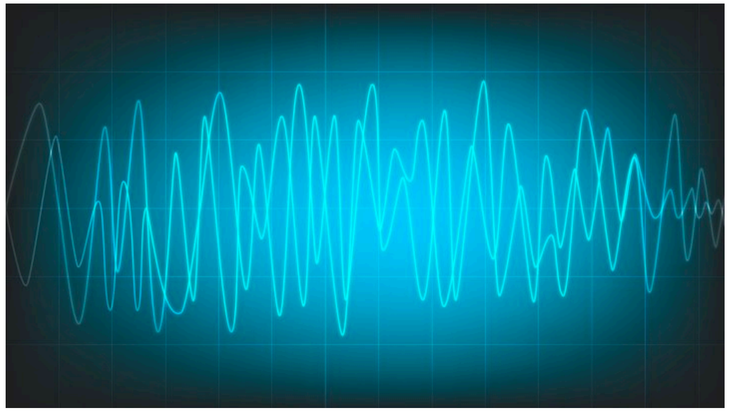
# Challenge: we need faster methods for optimization

# 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 $\theta$ ⟶

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

⟶ Optimal solution $z^\star(\theta)$

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

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

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

# Learning to Optimize

# The learning to optimize paradigm

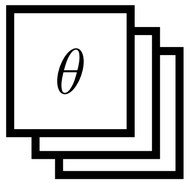
**Goal**: solve the parametric optimization problem fast

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

**Offline**

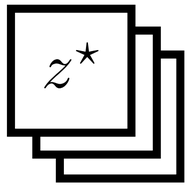
**Data collection**

Parameters




Solve

Optimal solutions




**Training**

Training parameter $\theta$

Learnable Optimizer with weights $w$

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

Loss

Learn

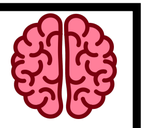
Candidate solution

Deploy

**Online evaluation**

Unseen parameter $\theta$

Learned Optimizer



High-quality solution

# Challenges in learning to optimize methods

I. Lack convergence guarantees

II. Lack generalization guarantees

III. Lack scalability

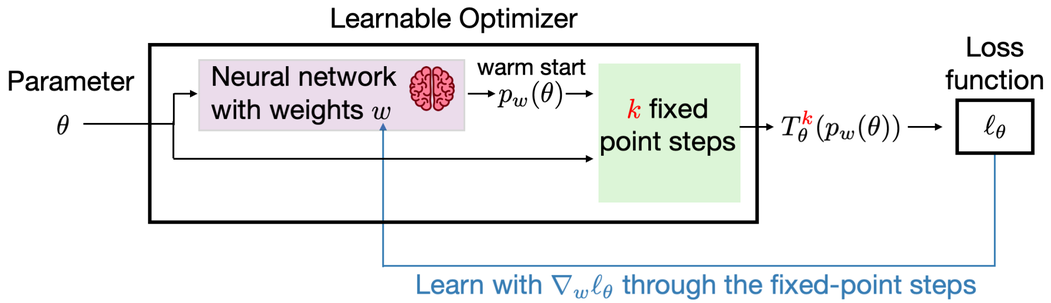
→ We need more **reliable** methods

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

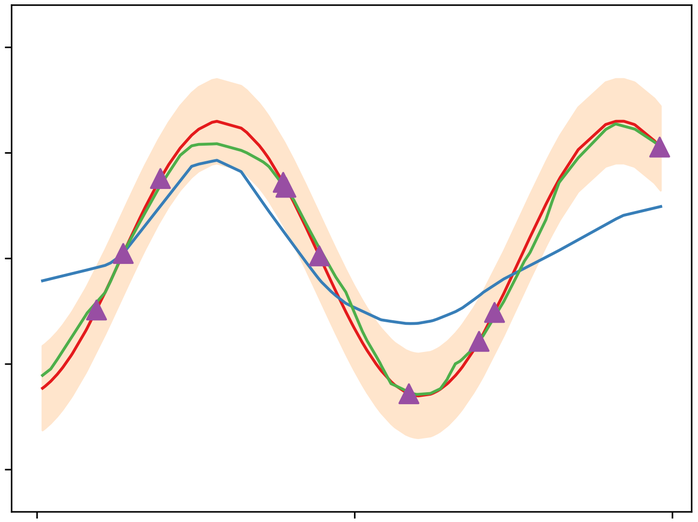
*"So, to conclude this article, let us quote Sir Winston Churchill: 'Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.'"*

# Talk Outline

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




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



# Collaborators

Georgina Hall

INSEAD

Brandon Amos

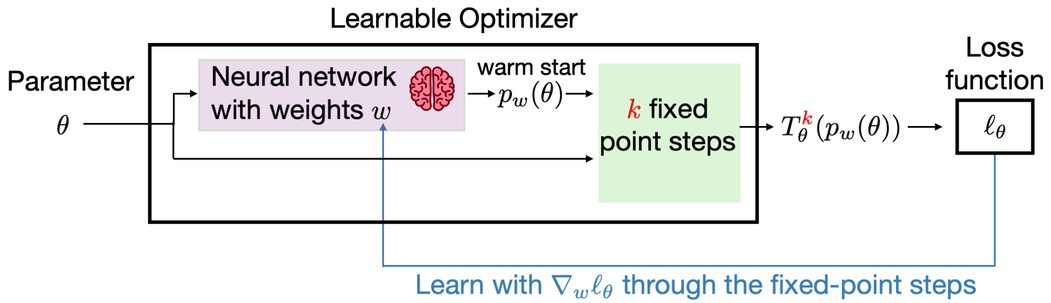
Meta

Bartolomeo Stellato

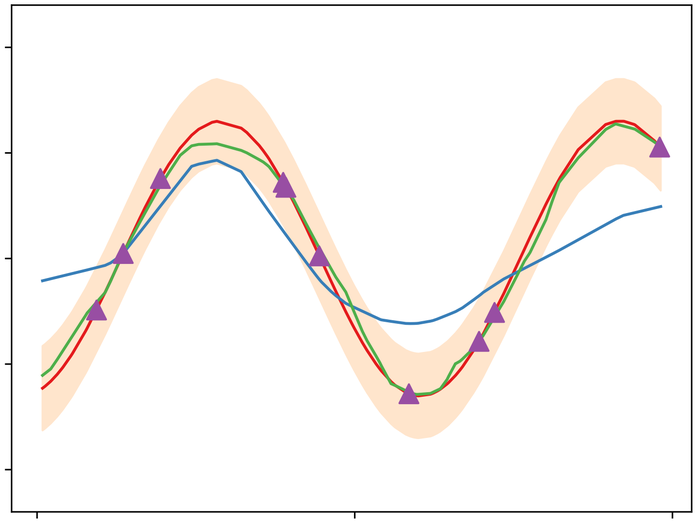
PRINCETON UNIVERSITY

# Talk Outline

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




- **Part 2: Data-Driven 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

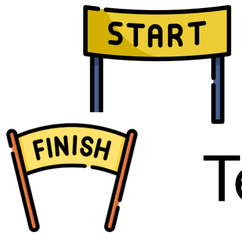
minimize $f_\theta(z)$

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 $\|T_\theta(z^i) - z^i\|_2$ is small **Fixed-point residual**

**Example: Proximal gradient descent**

$$\text{minimize} \quad \underset{\text{Convex}\atop\text{Smooth}}{g_\theta(z)} + \underset{\text{Convex}\atop\text{Non-smooth}}{h_\theta(z)}$$

**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**

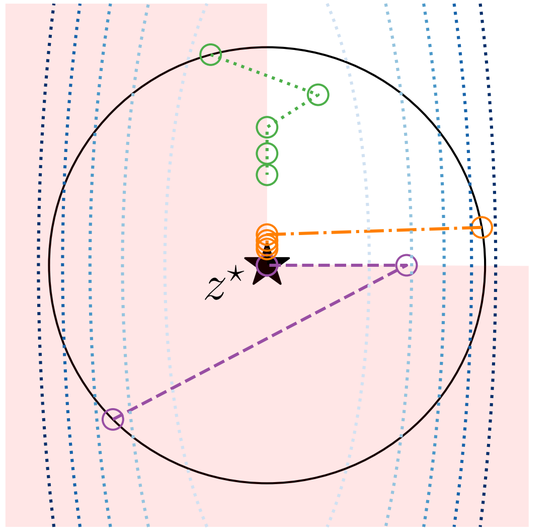
# Some warm starts are better than others

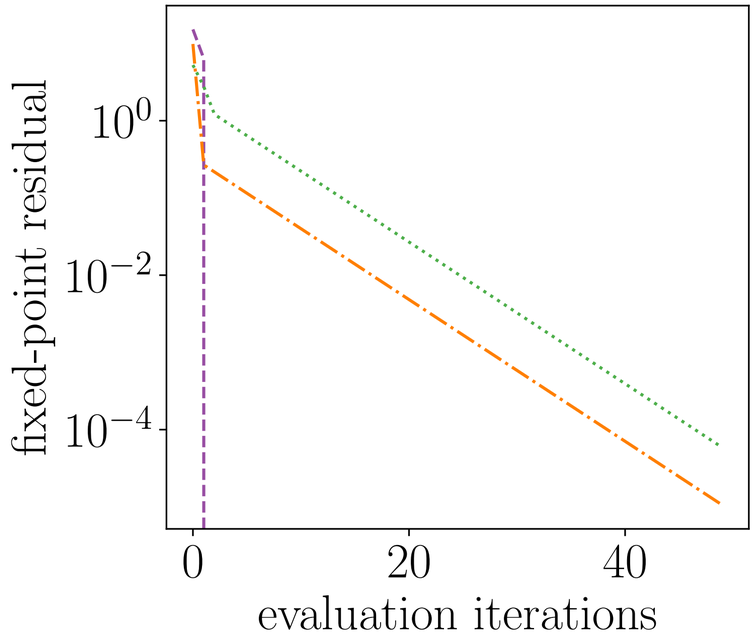
minimize $\quad 10z_1^2 + z_2^2$

subject to $\quad z \geq 0$

★ Optimal solution at the origin

Run proximal gradient descent to solve

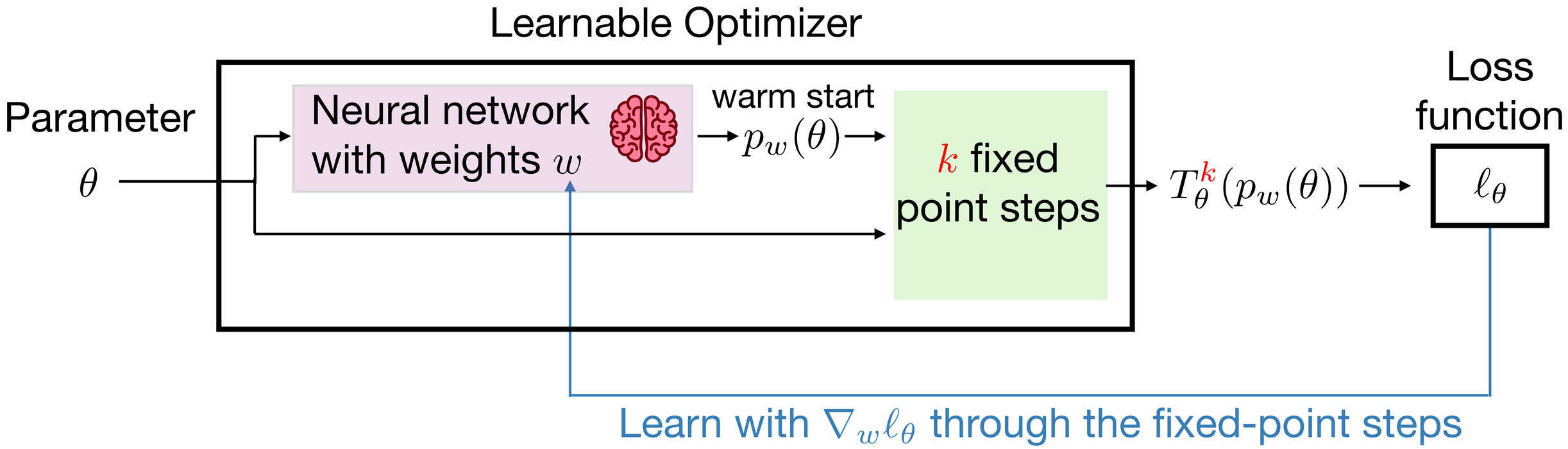



All three warm starts appear to be equally suboptimal but converge at very different rates

The quality of the warm start depends on the algorithm

# End-to-end learning architecture

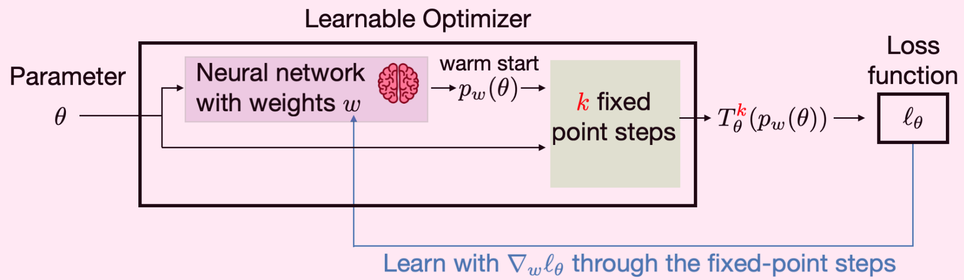



**Loss function:** $\ell_\theta(z) = \|z - z^\star(\theta)\|_2$ Ground truth solution

Learned warm start tailored for downstream algorithm

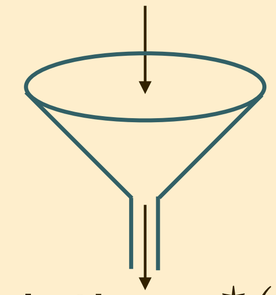
# Benefits of our learning framework

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




## Guaranteed convergence

Learned solver with convergence




## 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

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

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

Allows us to quantify solve time in seconds

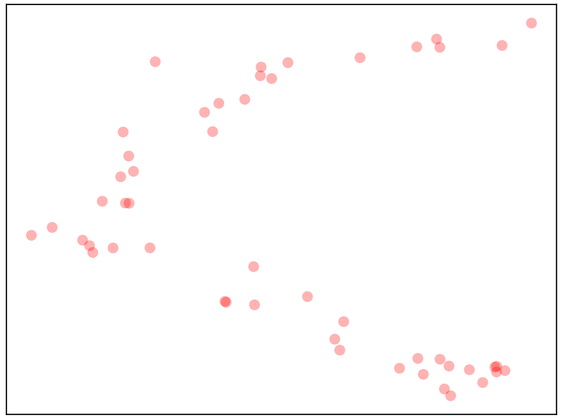
# Numerical Experiments

**Comparing our learned warm starts**  **against**

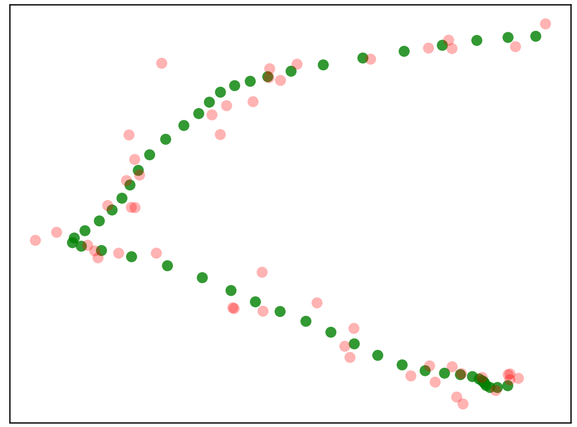
**Baseline initializations**

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

**Second-order cone program**

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

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

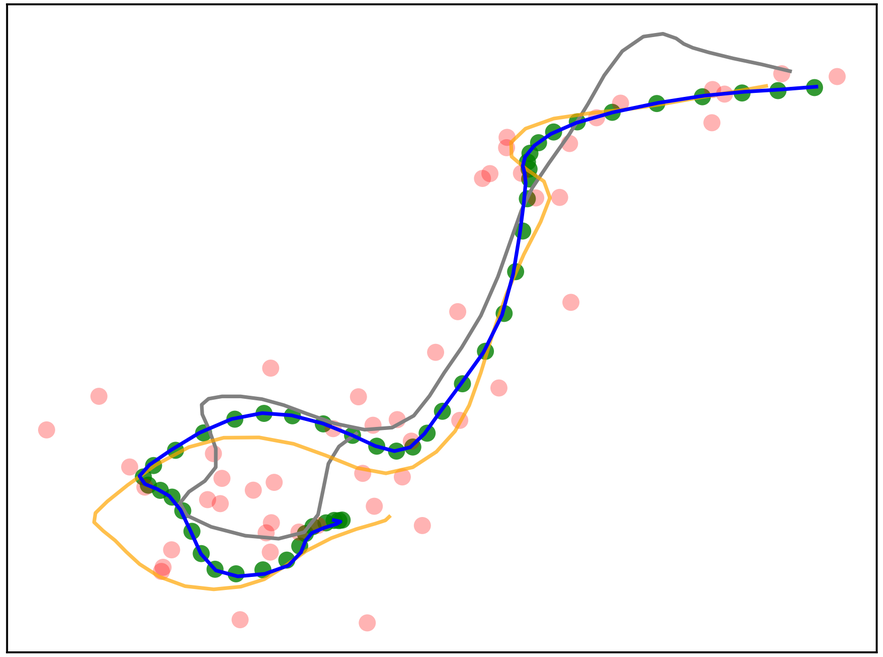
Recovered trajectory

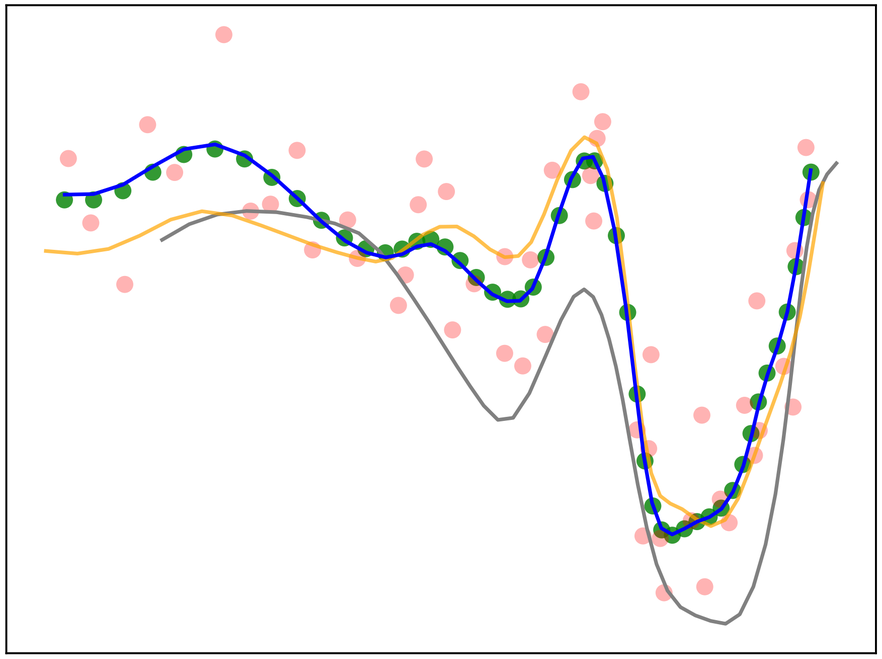
Dynamics matrices: $A, B$

Observation matrix: $C$

Huber loss: $\psi_\rho$

# 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 → **Controller** → Control inputs

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

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

$u_{\text{min}} \leq u_t \leq u_{\text{min}}$

$x_{\text{min}} \leq x_t \leq x_{\text{max}}$

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

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

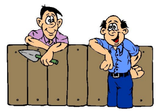
$u_{-1} = u_{\text{prev}}$

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

Linearized dynamics

# MPC of a quadcopter in a closed loop

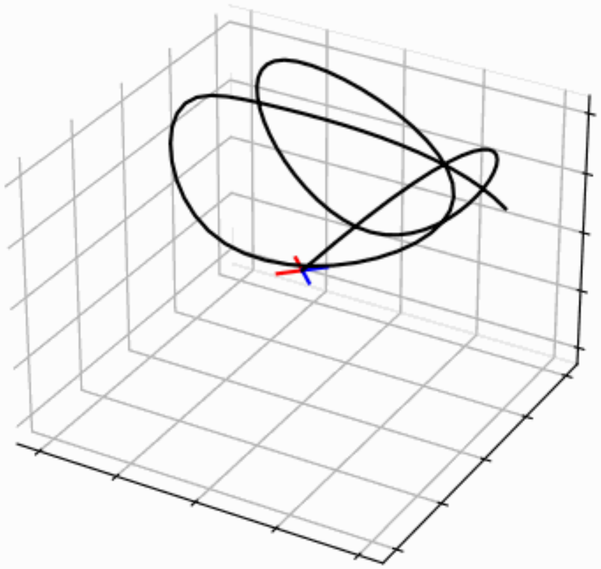
**Budget of 15 fixed-point steps**

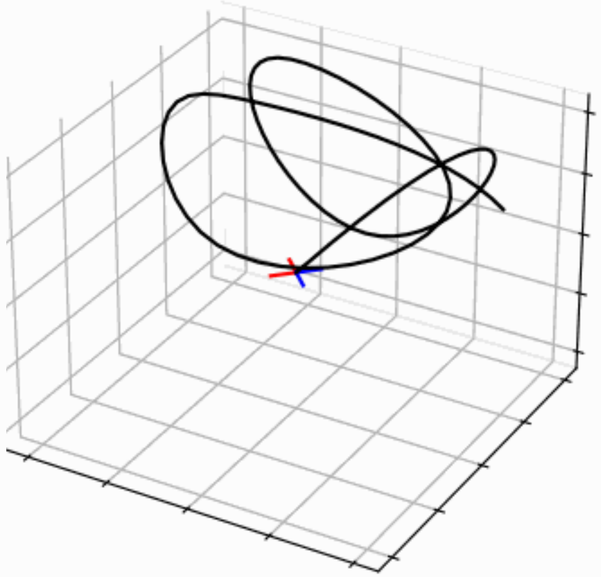


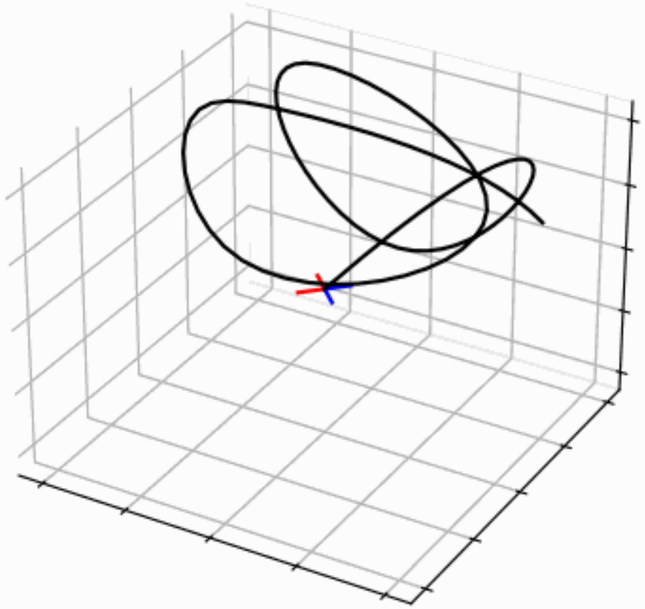
Nearest neighbor

Previous solution

Learned: $k = 5$







With learning, we can track the trajectory well

# Image deblurring

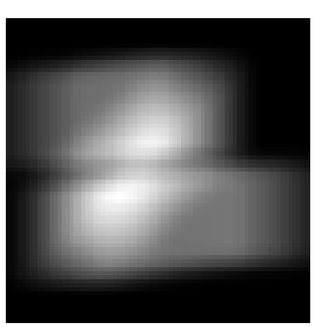


**Image deblurring**

$\theta = b$
Blurred image

**Quadratic program**

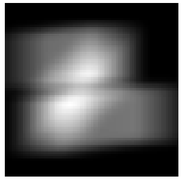
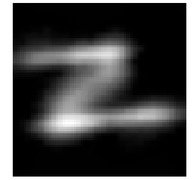
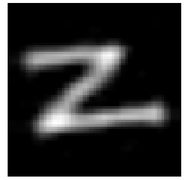
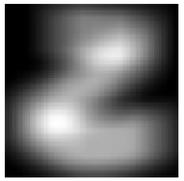
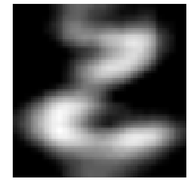
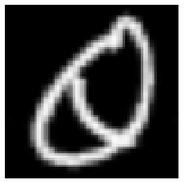
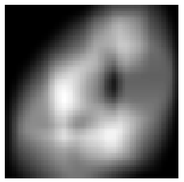
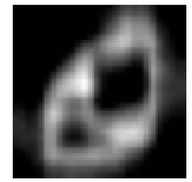
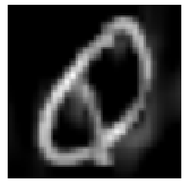
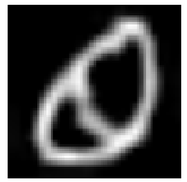
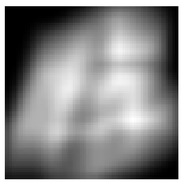
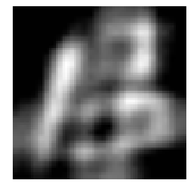
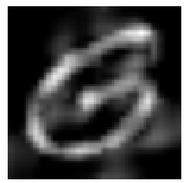
minimize $\quad \|Ax - b\|_2^2 + \lambda \|x\|_1$

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

$x^\star$
Deblurred image

$A$: blur operator

# Image deblurring

| percentile | optimal | blurred | cold-start | nearest neighbor | learned |
|---|---|---|---|---|---|
| $10^{\text{th}}$ |  |  |  |  |  |
| $50^{\text{th}}$ |  |  |  |  |  |
| $90^{\text{th}}$ |  |  |  |  |  |
| $99^{\text{th}}$ |  |  |  |  |  |

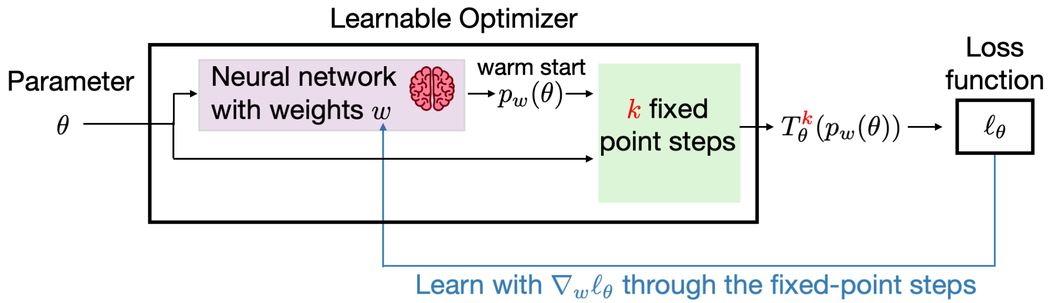
**50 fixed-point steps**

**Distance to nearest neighbor increases**

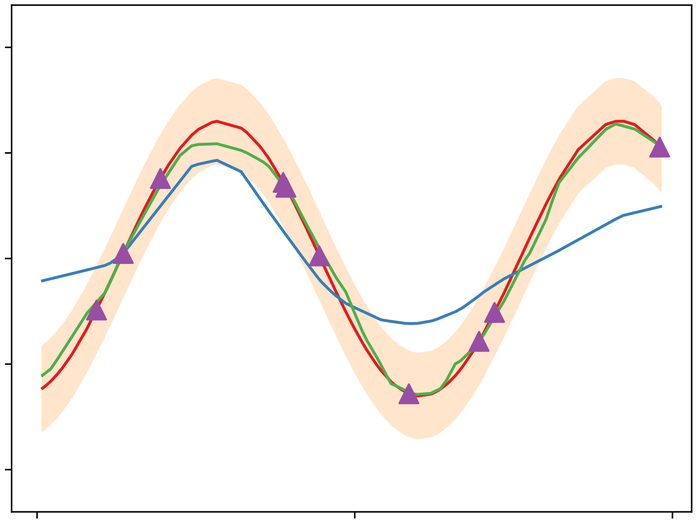
With learning, we can deblur all of the images quickly

# Talk Outline

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

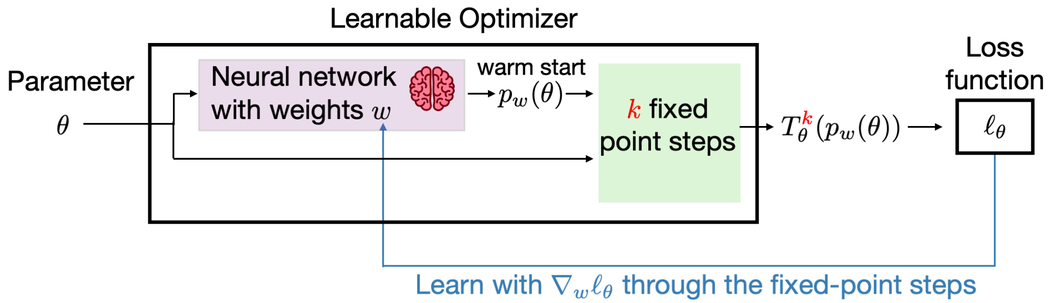



- **Part 2: Data-Driven Performance Guarantees for Classical and Learned Optimizers**



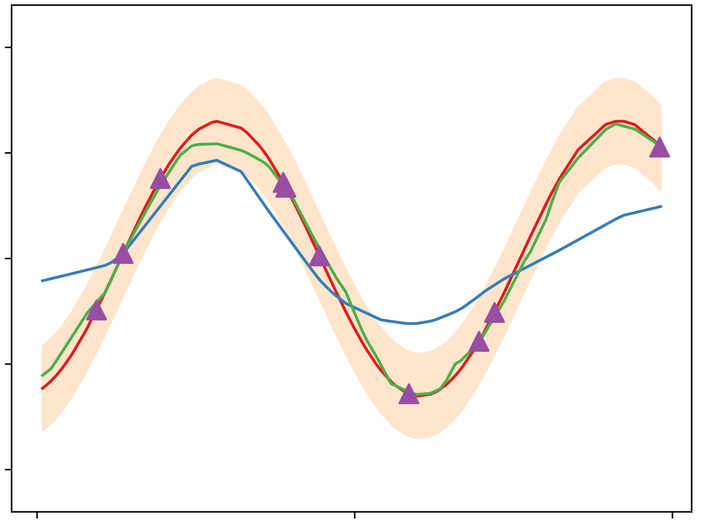
# Talk Outline

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



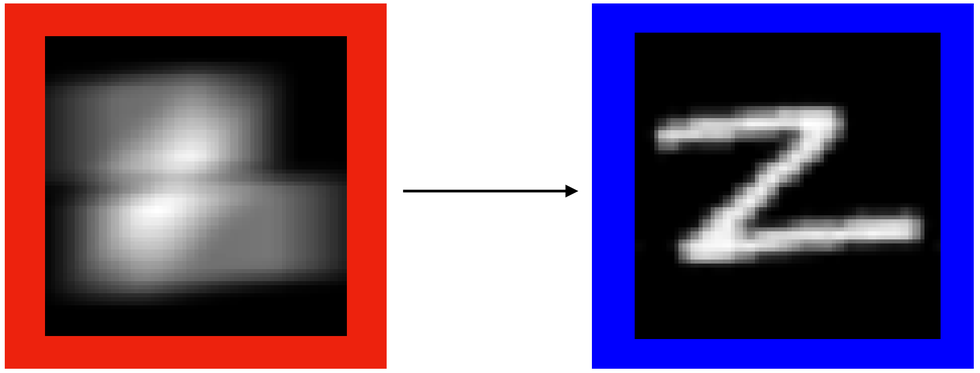

- **Part 2: Data-Driven Performance Guarantees for Classical and Learned Optimizers**

**Classical = no learning**



# Worst-case bounds can be very loose

Example: image deblurring

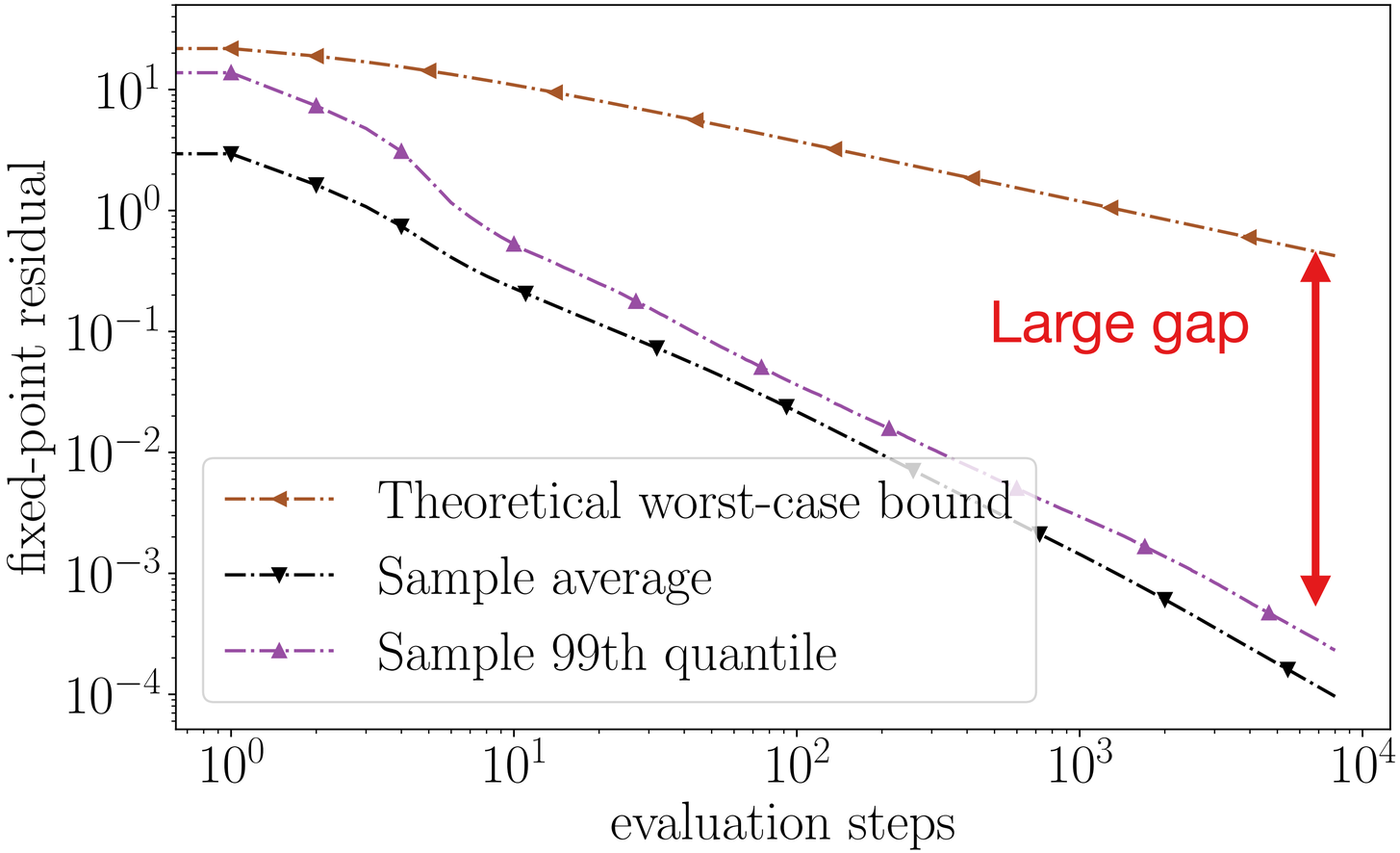


**Quadratic program**

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

subject to $0 \leq x \leq 1$

1000 problems solved with




Worst-case bounds are pessimistic and do not consider the parametric structure

Approach: probabilistically bound over the parametric distribution

# Our recipe for guarantees for classical optimizers

algorithm steps

tolerance

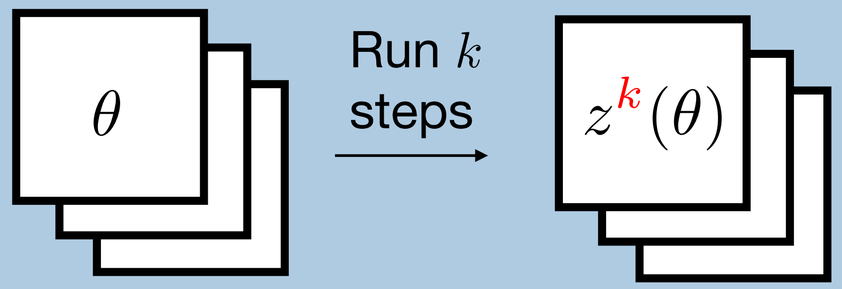
$$e(\theta) = \mathbf{1}(\ell^k(\theta) > \epsilon)$$

Any metric
(*e.g.*, fixed-point residual)

Step 1
Run $k$ steps
for $N$ parametric problems

Parameters

Candidate solutions




Step 2
Evaluate the empirical risk

$$\frac{1}{N}\sum_{i=1}^{N} e(\theta_i)$$

Step 3
Bound the risk
(Next slide)

$$\text{risk} = \mathbf{E}_{\theta\sim\mathcal{X}} e(\theta) \leq \text{bound}$$

# 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]

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

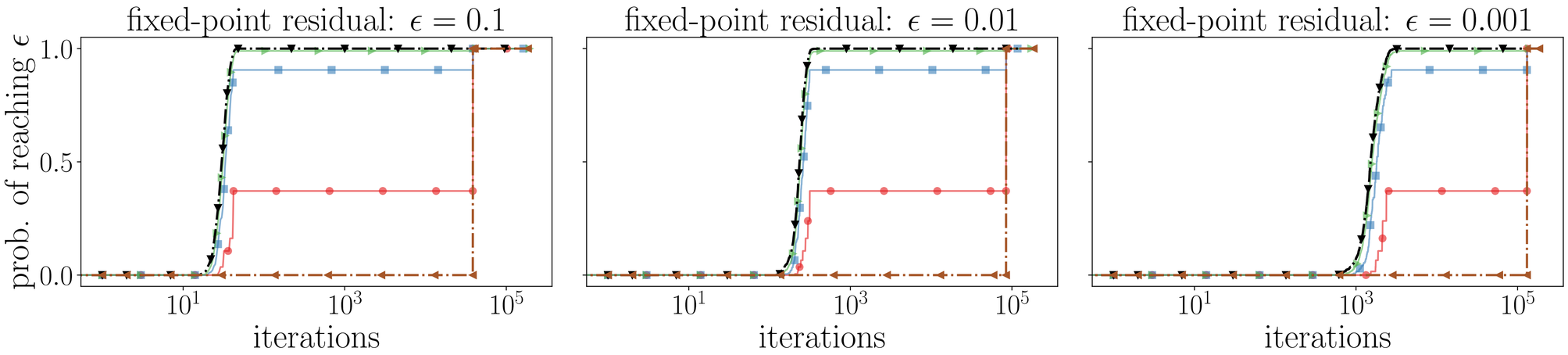
$$\mathbf{P}(\ell^k(\theta) > \epsilon) = \text{risk} \leq \mathrm{KL}^{-1}\,(\text{empirical risk} \mid \text{regularizer})$$

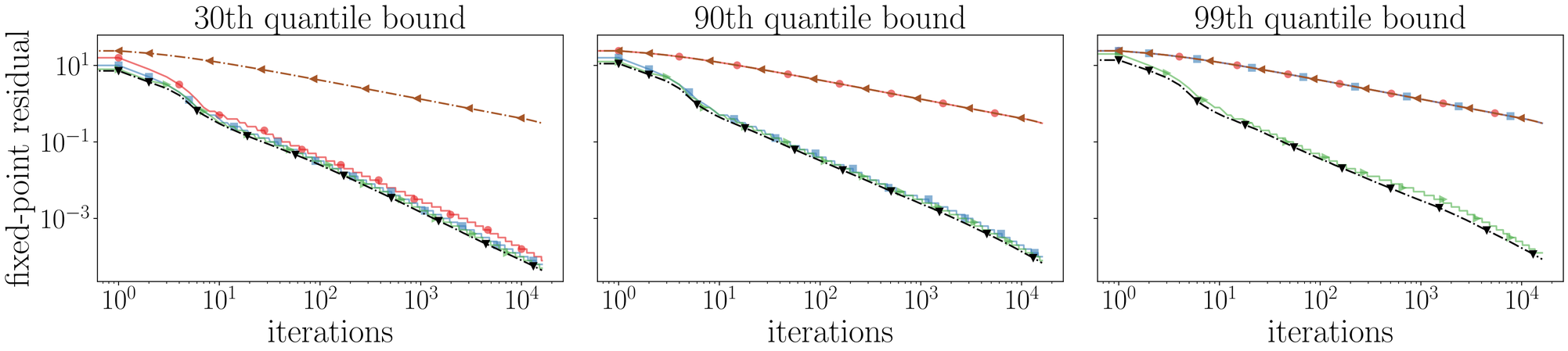
Number of problems

1D convex optimization problem

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

# Image deblurring guarantees





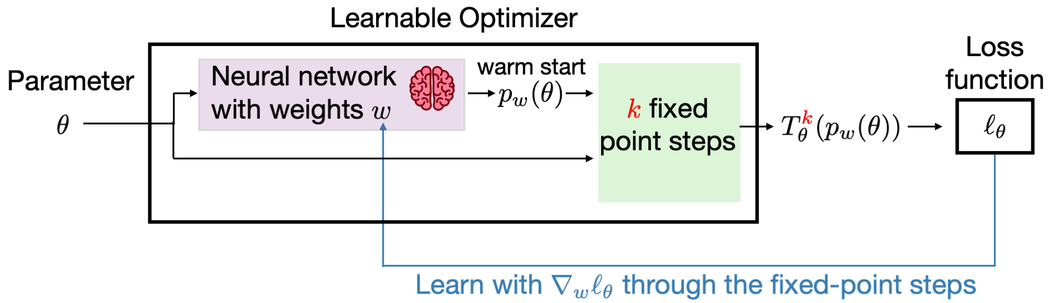

Probabilistic bound with



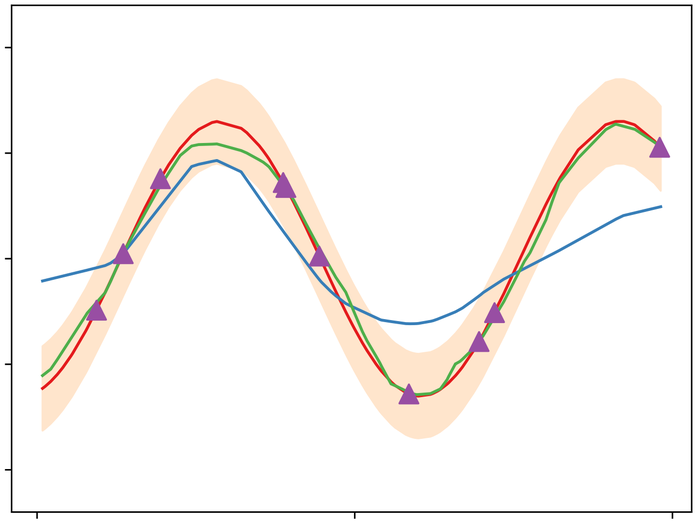

With 1000 samples, we provide strong probabilistic guarantees on the 99th quantile

# Talk Outline

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




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



**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 $1 - \delta$

$$\mathbf{E}_{\theta\sim\mathcal{X}}\mathbf{E}_{w\sim P}e_w(\theta) \leq \mathrm{KL}^{-1}\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{E}_{w\sim P}e_w(\theta_i)\middle|\frac{1}{N}\left(\mathrm{KL}(\mathrm{P}\parallel\mathrm{P}_0)+\log(\mathrm{N}/\delta)\right)\right)$$

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

Optimize the bounds directly

# 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, $\sigma$: noise

Standard technique

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

ISTA (iterative shrinkage thresholding algorithm)

(Classical optimizer)

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

Learned ISTA

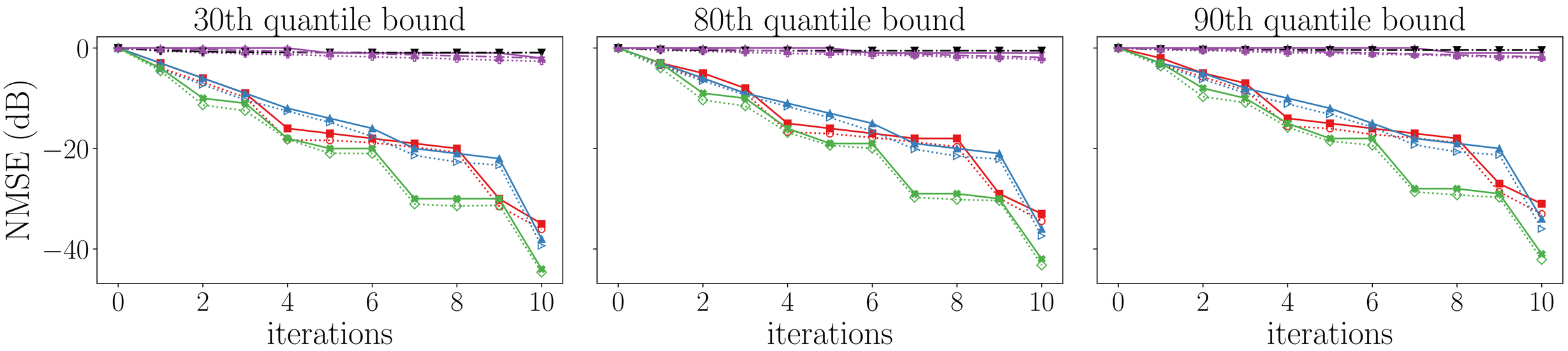
(Learned optimizer)

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

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

$\text{soft threshold}_\psi(z) = \textbf{sign}(z)\max(0, |z| - \psi)$

# Learned ISTA results for sparse coding

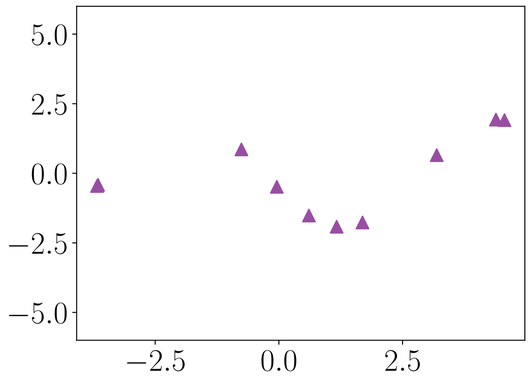



| Not learned | Learned | | | |
|---|---|---|---|---|
| ISTA | LISTA | ALISTA | TiLISTA | GLISTA |
| Bound | ★ | ▲ | ■ | ✖ |
| Empirical | ▼ | ✚ | ▷ | ○ | ◇ |

Our bounds are close to empirical performance

Learned optimizers provably perform well in just 10 steps

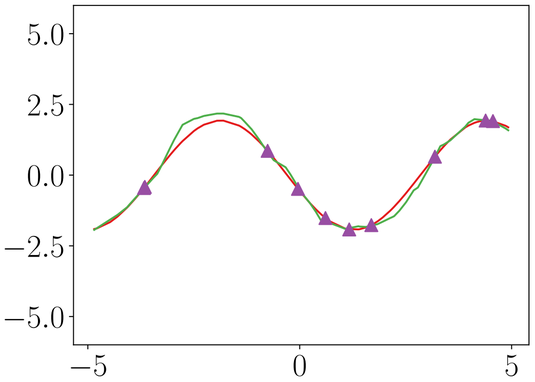
# K-shot Meta-Learning for Sine Curves




**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**

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

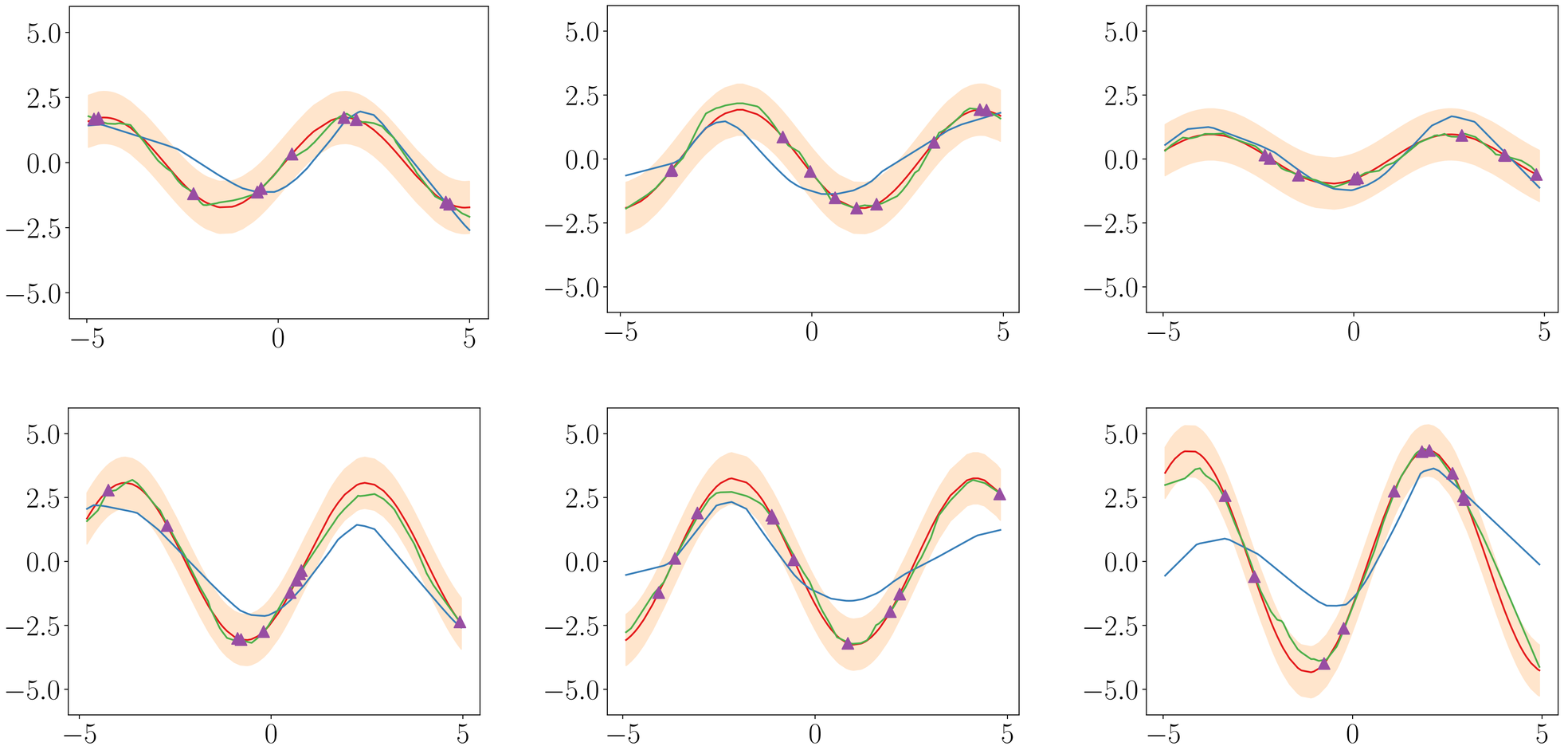
Weights that generalize to new points quickly

$\hat{z}$

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




Used for gradients

Ground truth

After 10 grad steps

Empirical MAML

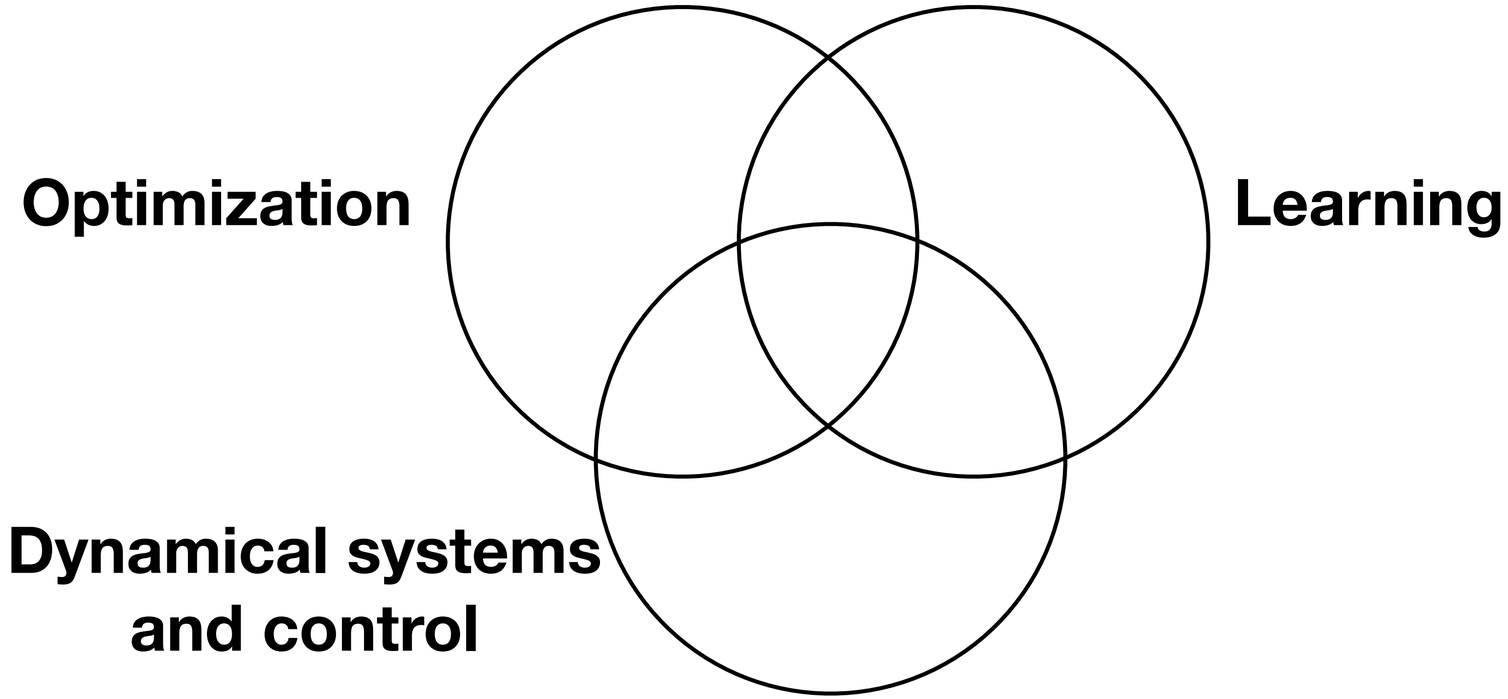
Pretrained

Region with MAML guarantee

With high probability, 87% of the time MAML stays within the band

The pretrained baseline only stays within the band 33% of the time

# Future directions



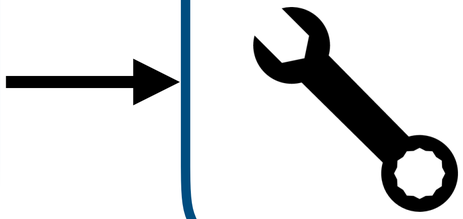

**Areas of interest**

- Learning dynamical systems
- Learning safety and stability certificates
- Guarantees via statistical learning

# Conclusions: Learning Optimizers with Guarantees

**Traditional view**

- General purpose
- One-size fits all
- With guarantees



**New view**

- Task-specific
- Trainable
- With guarantees

**Data** can help us

- **Accelerate** optimization algorithms
- **Guarantee** their performance

**Data-Driven Performance Guarantees for Classical and Learned Optimizers**
R. Sambharya, B. Stellato
*Submitted: Journal of Machine Learning Research*
https://arxiv.org/pdf/2404.13831.pdf

**Learning to Warm-Start Fixed-Point Optimization Algorithms**
R. Sambharya, G. Hall, B. Amos, B. Stellato
*Journal of Machine Learning Research (to appear)*
https://arxiv.org/pdf/2309.07835.pdf

rajivs@princeton.edu

rajivsambharya.github.io