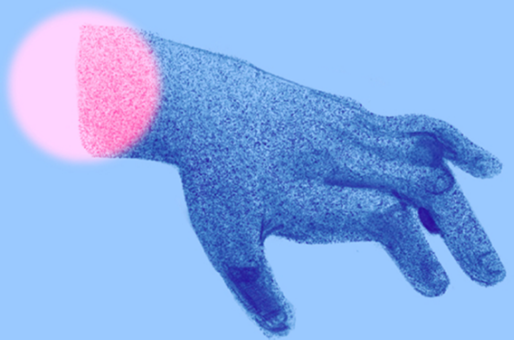
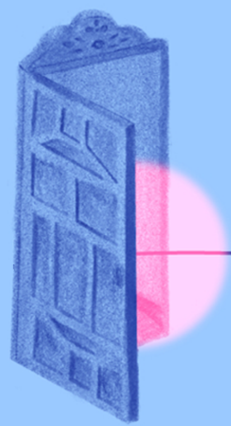
CONSTRAINED OPTIMIZATION *for* MACHINE LEARNING





Jose Gallego-Posada

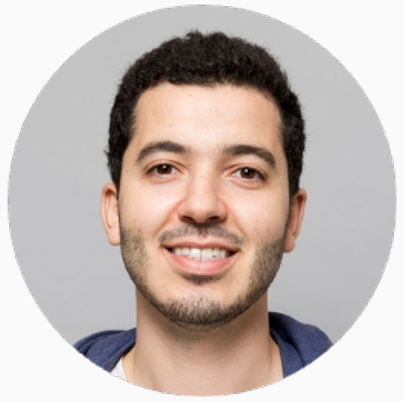
August 15, 2024

Today's agenda

- Why constrained optimization?
- Foundations of constrained optimization
- Three core papers
- Conclusions and perspectives

"If I had been rich, I probably would not have devoted myself to mathematics."

Collaborators



Akram Erraqabi

Tianyue H. Zhang

Juan Ramirez

Meraj Hashemizadeh

Motahareh Sohrabi

Rohan Sukumaran

Simon Lacoste-Julien

Golnoosh Farnadi

Yoshua Bengio

Mila

Université de Montréal

CIFAR

IVADO

Contributions

Controlled Sparsity via Constrained Optimization
Gallego-Posada, *Ramirez, Erraqabi, Bengio, Lacoste-Julien*
NeurIPS 2022

L0onie: Compressing COINs with L0-constraints
Ramirez, ***Gallego-Posada****
Sparsity in Neural Networks Workshop 2022

Balancing Act: Constraining Disparate Impact in Sparse Models
Hashemizadeh, Ramirez*, Sukumaran, Farnadi, Lacoste-Julien,* ***Gallego-Posada***
ICLR 2024

On PI controllers for updating Lagrange multipliers in constrained optimization
Sohrabi, Ramirez*, Zhang, Lacoste-Julien,* ***Gallego-Posada***
ICML 2024

Cooper: A Library for Constrained Optimization in Deep Learning
Gallego-Posada*, *Ramirez*, Hashemizadeh, Lacoste-Julien*
JMLR MLOSS 2024 (under submission)

Excluded Contributions

GAIT: A Geometric Approach to Information Theory
Gallego-Posada, Vani, Schwarzer, Lacoste-Julien
AISTATS 2020

Equivariant Mesh Attention Networks
Basu, Gallego-Posada*, Vigano*, Rowbottom*, Cohen*
TMLR 2022

AI & Cities: Risks, Applications and Governance
Koseki, Jameson et al.
Tech Report - Mila and UN-Habitat 2022

A Distributed Data-Parallel PyTorch Implementation of the Distributed Shampoo Optimizer for Training Neural Networks At-Scale
Shi, Lee, Iwasaki, Gallego-Posada, Li, Rangadurai, Mudigere, Rabbat
Tech Report 2024

Introduction

Widespread deployment of powerful machine learning models has resulted in mounting pressures to enhance the robustness, safety and fairness of such models—often arising from regulatory and ethical considerations

~~“Build now, fix later”~~

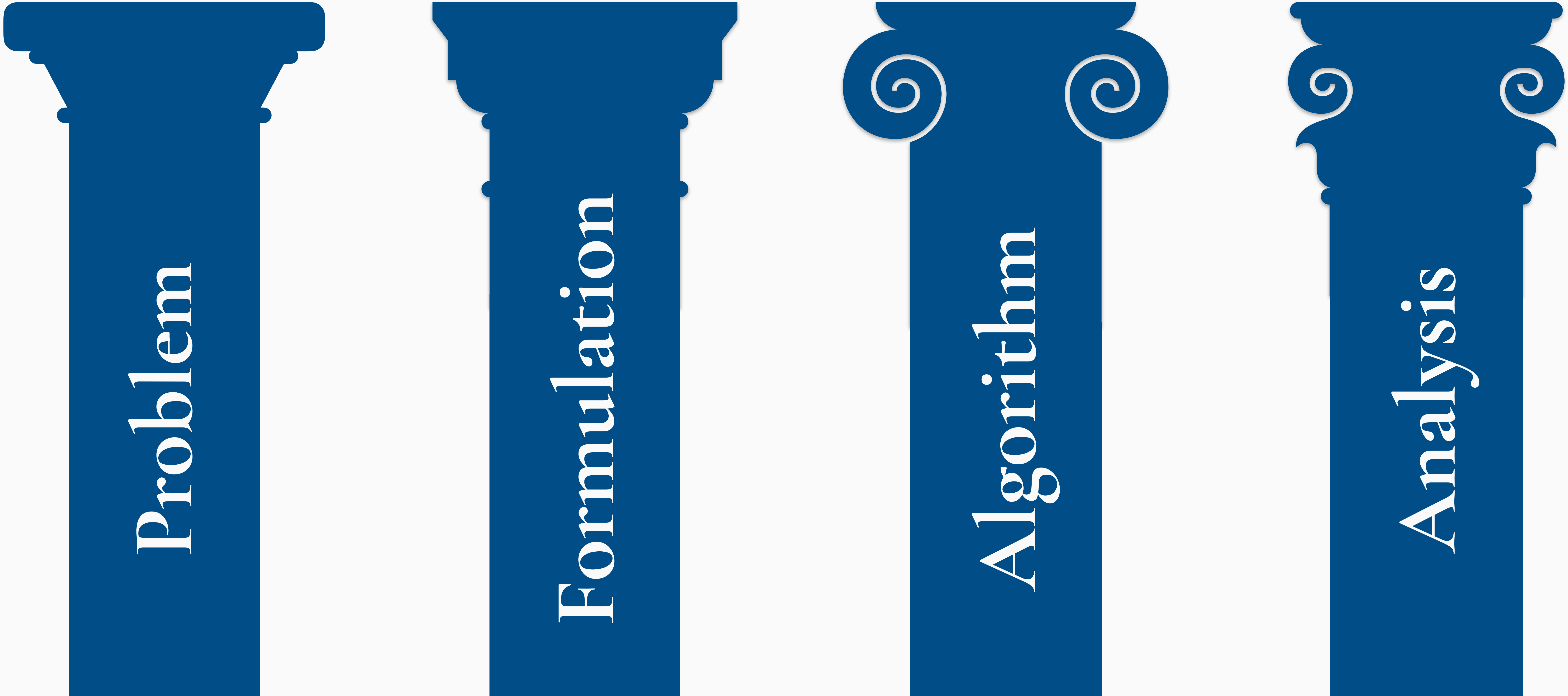
- **Inability to guarantee compliance** with industry standards and governmental regulations **limits implementation** of ML solutions in real-world applications
- Retro-fitting safety measures as afterthoughts!
- Continuous **incurrence of technical debt hinders long-term progress** of the field

Secure by design

- We advocate for a paradigm shift in which **constraints are an integral part** of the model development process
- Constrained optimization offers a **rich conceptual framework** accompanied by **algorithmic tools** for reliably enforce complex properties on ML models

Recent works on CO for ML

- **Fairness:** Zafar et al. (2017); Cotter et al. (2019); Hashemizadeh et al. (2024)
- **Safe reinforcement learning:** Stooke et al. (2020)
- **Sparse neural network training:** Gallego-Posada et al. (2022)
- **Active learning:** Elenter et al. (2022)
- **Model quantization:** Hounie et al. (2023)
- **Dynamics of constrained learning:** Sohrabi et al. (2024)
- **Safe RLHF:** Dai et ail. (2024)


Problem
Formulation
Algorithm
Analysis

Problem

Formulation

Algorithm

Analysis

- Controllability
- Hyperparameter interpretability
- Better exploration of trade-offs
- Experimental accountability

Problem

Formulation

Algorithm

Analysis

- Game structure
- Functional representation of the problem

- Learning dynamics
- Computational cost
- Convergence guarantees
- Practical robustness

Problem

Formulation

Algorithm

Analysis

- Feasibility reigns
- Two axis for generalization
- How fast to become feasible?

  How fast to improve the loss?

Constrained optimization

$$\underset{\boldsymbol{x}}{\text{minimize}}\ f(\boldsymbol{x})$$

subject to $\boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0}_m$ and $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}_n$

Feasible set

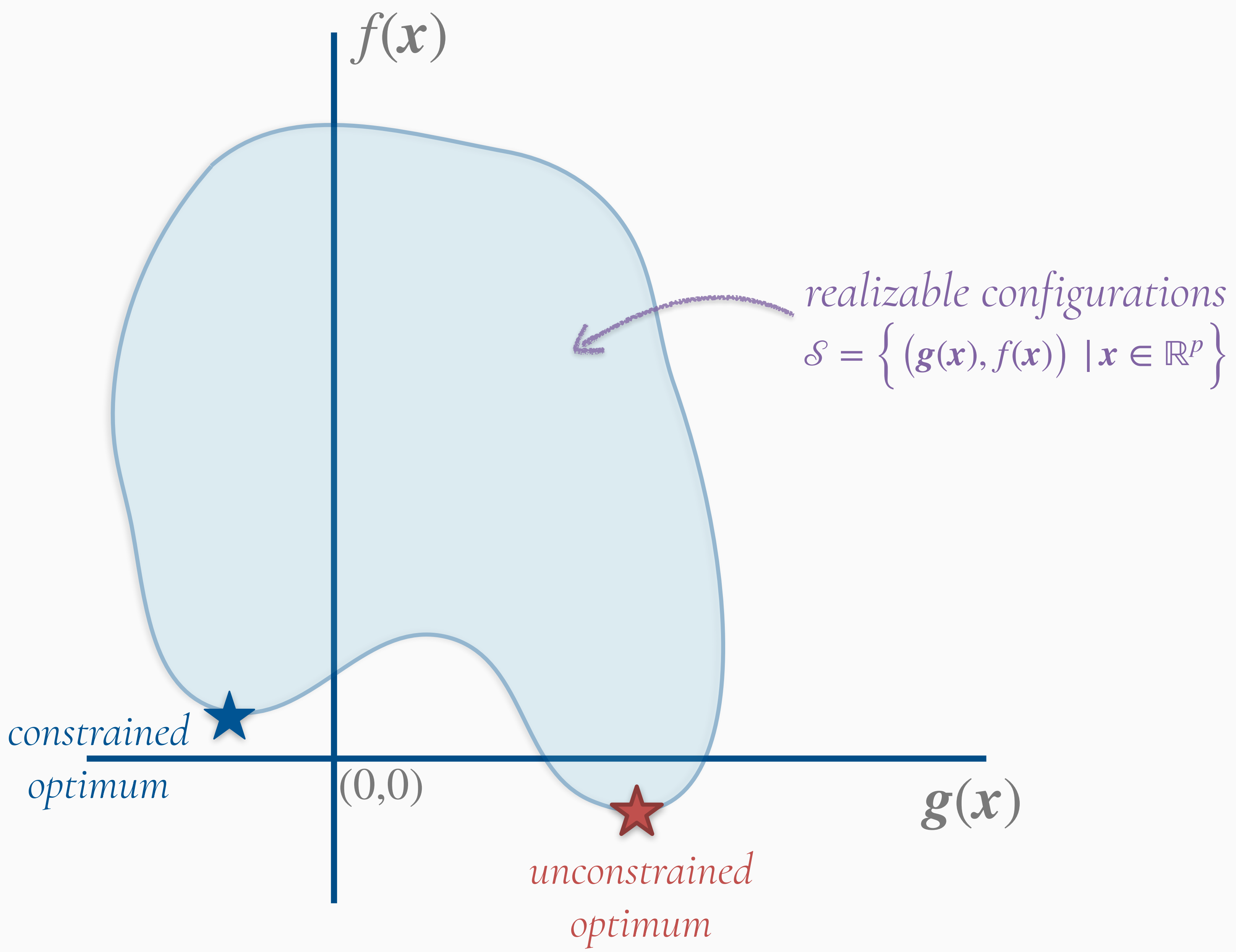
$$\Omega = \left\{\boldsymbol{x} \in \mathbb{R}^p \,|\, \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0} \text{ and } \boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0}\right\}$$

Optimality condition (necessary)

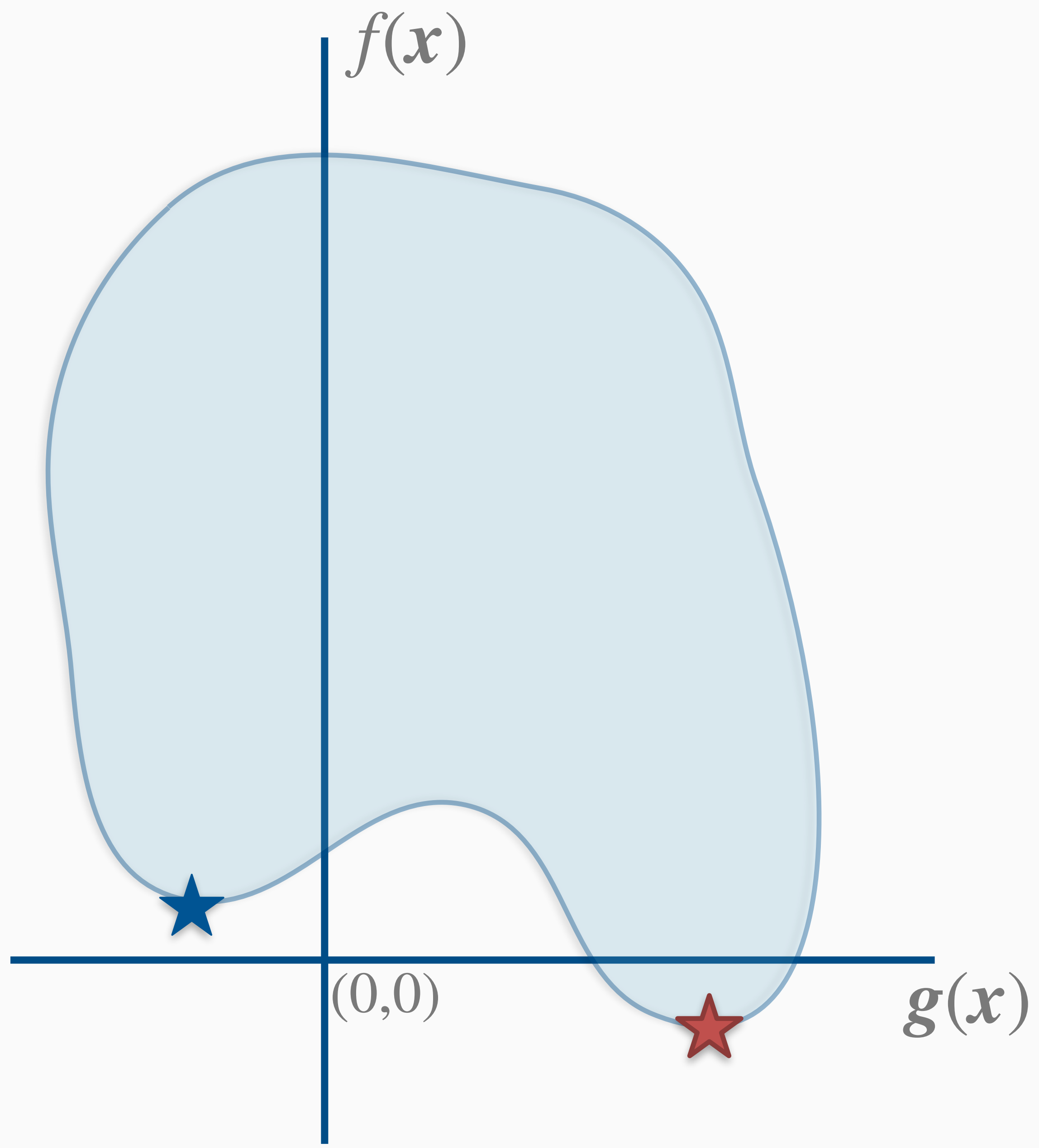
If $\boldsymbol{x}^*$ is a local minimum of f over Ω, then

$$\nabla f(\boldsymbol{x}^*)^\top z \geq 0 \quad \forall z \in \mathrm{FD}(\boldsymbol{x}^*)$$

feasible directions at $\boldsymbol{x}^*$



Feasibility and accountability

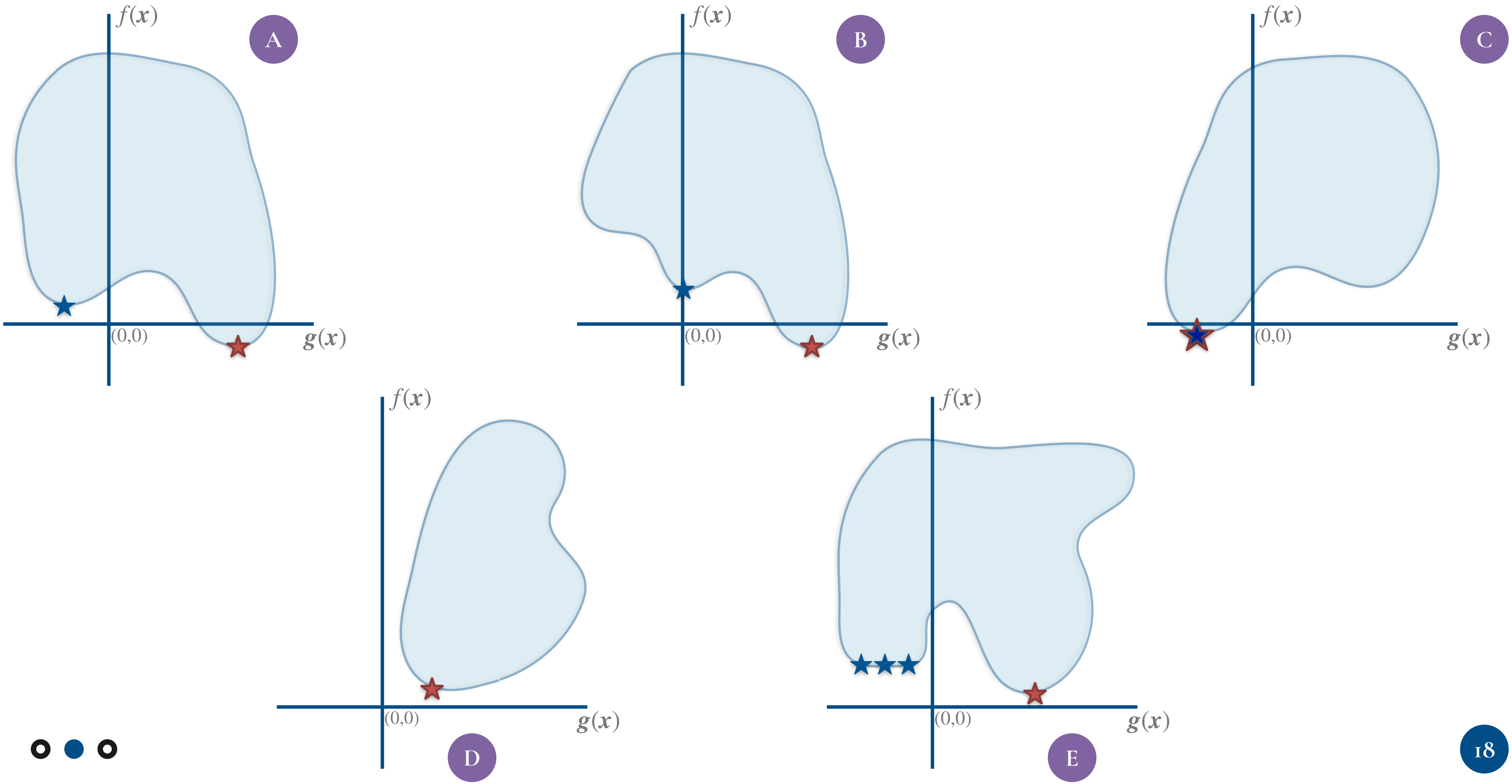


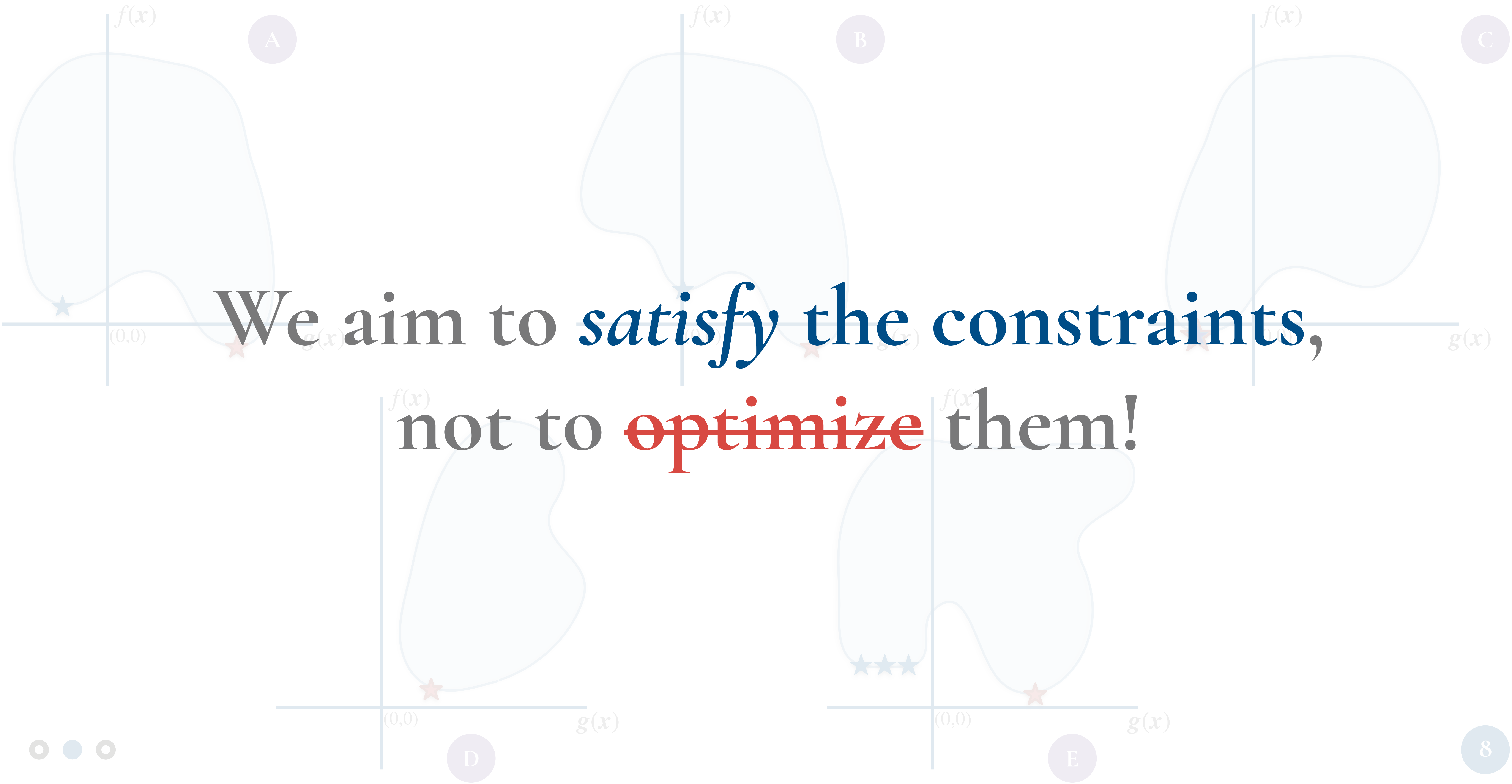
Although in the unconstrained setting ★ would be preferred, not valid solution for constrained problem since it is infeasible.

usually informed by problem-dependent requirements

Choosing the constraint level **beforehand** ensures experimental accountability.

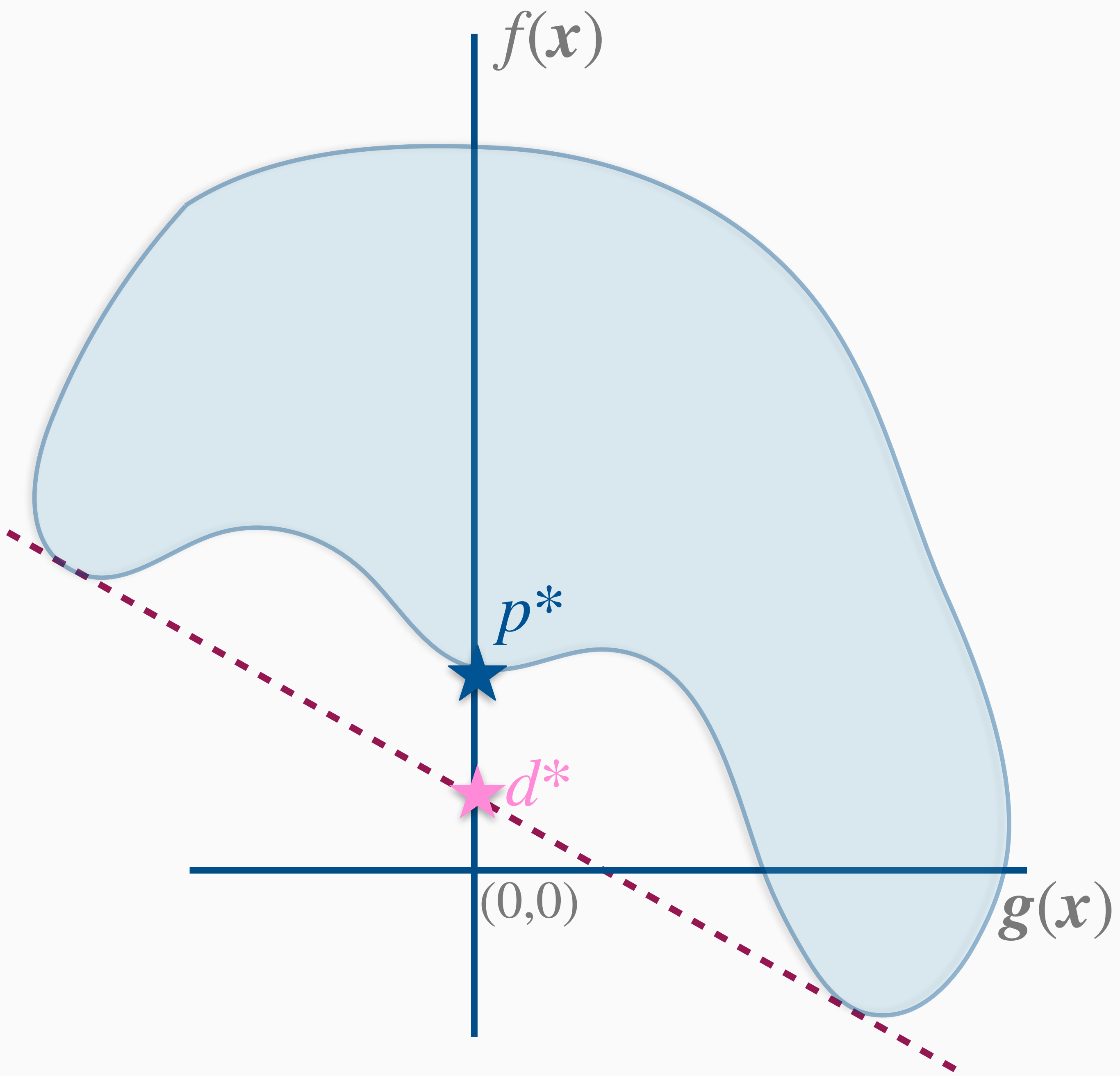
"not allowed to cheat"


A
B
C
D
E
f(x)
g(x)
(0,0)



We aim to *satisfy* the constraints, not to ~~optimize~~ them!

Why not just penalize?



Duality gap $\triangleq p^* - d^* \geq 0$

tunable hyperparameter

$$\underset{\boldsymbol{x}}{\text{minimize}}\ f(\boldsymbol{x}) + \lambda_{\text{pen}}\, g(x)$$

Tuning λ_{pen} typically requires a trial-and-error search!

In non-convex problems, there may be trade-offs between the objective and constraints that are not reachable using a penalized formulation.

Lagrangian problem

$$\min_{\boldsymbol{x}} f(\boldsymbol{x})$$

subject to $\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}_m$ and $\boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}_n$

$\Leftrightarrow$

Lagrangian

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda} \geq \mathbf{0}, \boldsymbol{\mu}} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \triangleq f(\boldsymbol{x}) + \boldsymbol{\lambda}^\top \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{\mu}^\top \boldsymbol{h}(\boldsymbol{x})$$

"Lagrange multipliers" or "dual variables"

Role of the multipliers (cf. Karush-Kuhn-Tucker necessary conditions)

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla \boldsymbol{g}_i(\boldsymbol{x}^*) + \sum_{i=1}^{n} \boldsymbol{\mu}_i^* \nabla \boldsymbol{h}_i(\boldsymbol{x}^*) = \mathbf{0}$$

Algorithmic approach

Saddle points of the Lagrangian correspond to constrained optima, but may not exist.

Find a min-max point!

Gradient Descent-Ascent (GDA)

Lagrangian $\quad \min_{x} \max_{\lambda \geq 0, \mu} \mathfrak{L}(x, \lambda, \mu) \triangleq f(x) + \lambda^\top g(x) + \mu^\top h(x)$

Algorithm

Initialize x_0, $\lambda_0 = \mathbf{0}$ and $\mu_0 = \mathbf{0}$

Repeat

$$\mu_{k+1} \leftarrow \mu_k + \eta_{\text{dual}} \nabla_{\mu} \mathfrak{L}(x_k, \lambda_k, \mu_k) = \mu_k + \eta_{\text{dual}} h(x_k)$$

$$\lambda_{k+1} \leftarrow \left[\lambda_k + \eta_{\text{dual}} \nabla_{\lambda} \mathfrak{L}(x_k, \lambda_k, \mu_k)\right]^+ = \left[\lambda_k + \eta_{\text{dual}} g(x_k)\right]^+$$

projected gradient ascent maintains non-negativity of inequality multipliers

$$x_{k+1} \leftarrow x_k - \eta_{\text{primal}} \nabla_{x} \mathfrak{L}(x_k, \lambda_k, \mu_k)$$

If convergence check satisfied; **stop**

Gradient Descent-Ascent (GDA)

$$\boldsymbol{\mu}_{k+1} \leftarrow \boldsymbol{\mu}_k + \eta_{\text{dual}} \nabla_{\boldsymbol{\mu}} \mathfrak{L}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k)$$

$$\boldsymbol{\lambda}_{k+1} \leftarrow \left[\boldsymbol{\lambda}_k + \eta_{\text{dual}} \nabla_{\boldsymbol{\lambda}} \mathfrak{L}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k)\right]^+$$

$$\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - \eta_{\text{primal}} \nabla_{\boldsymbol{x}} \mathfrak{L}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k)$$

pick your favourite primal optimizer!

Extensibility

Simplest possible first-order strategy. Can be combined with **more sophisticated updates.**

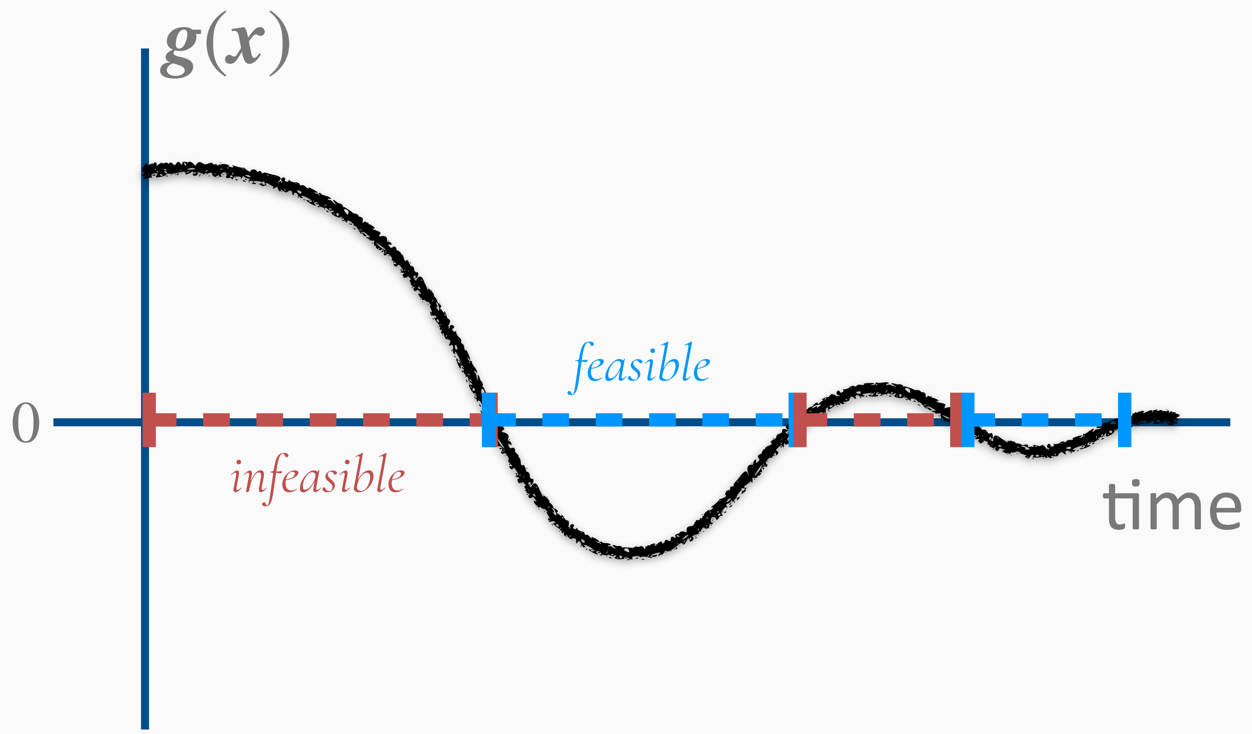
Negligible computational overhead

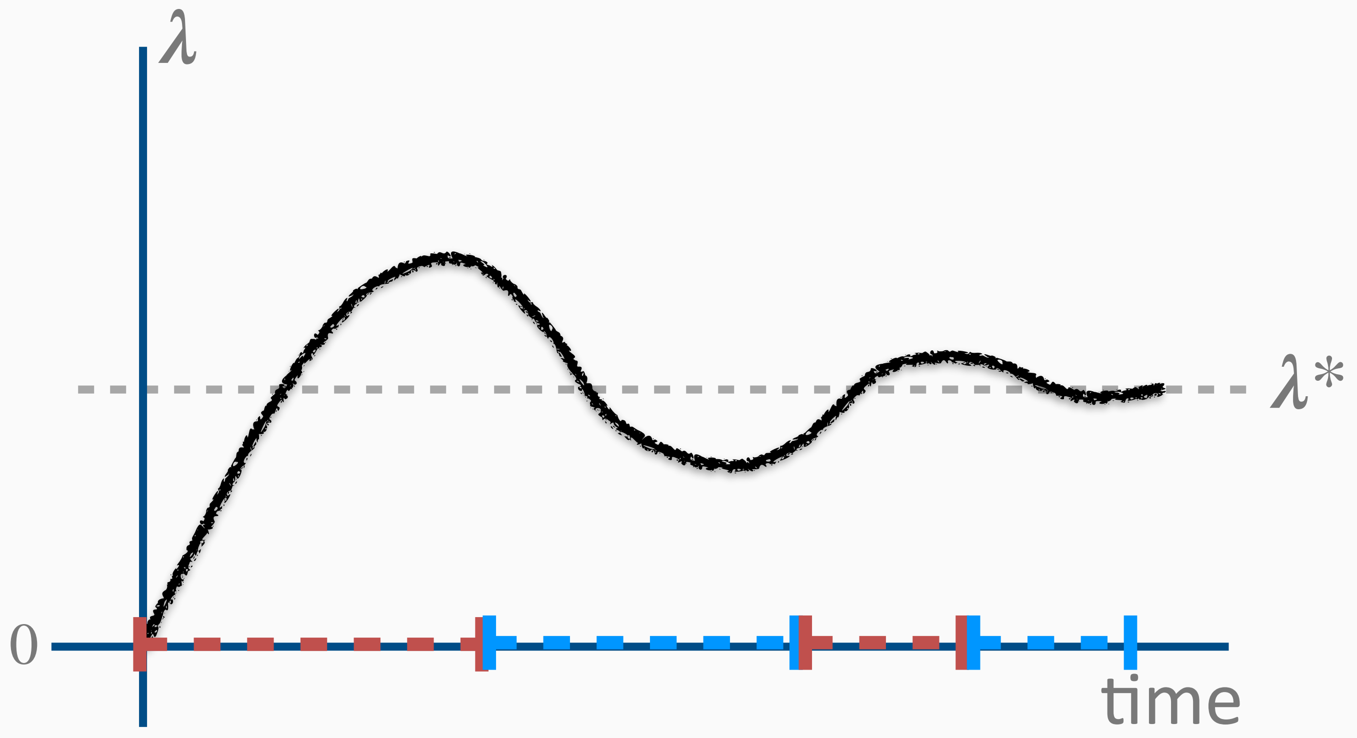
Compared to the penalized approach: only need to update value of the multipliers.

Dynamics of GDA

$$\lambda_{k+1} = \left[\lambda_k + \eta_{\text{dual}} \nabla_\lambda \mathfrak{L}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k)\right]^+ = \left[\lambda_k + \eta_{\text{dual}} \boldsymbol{g}(\boldsymbol{x}_k)\right]^+$$

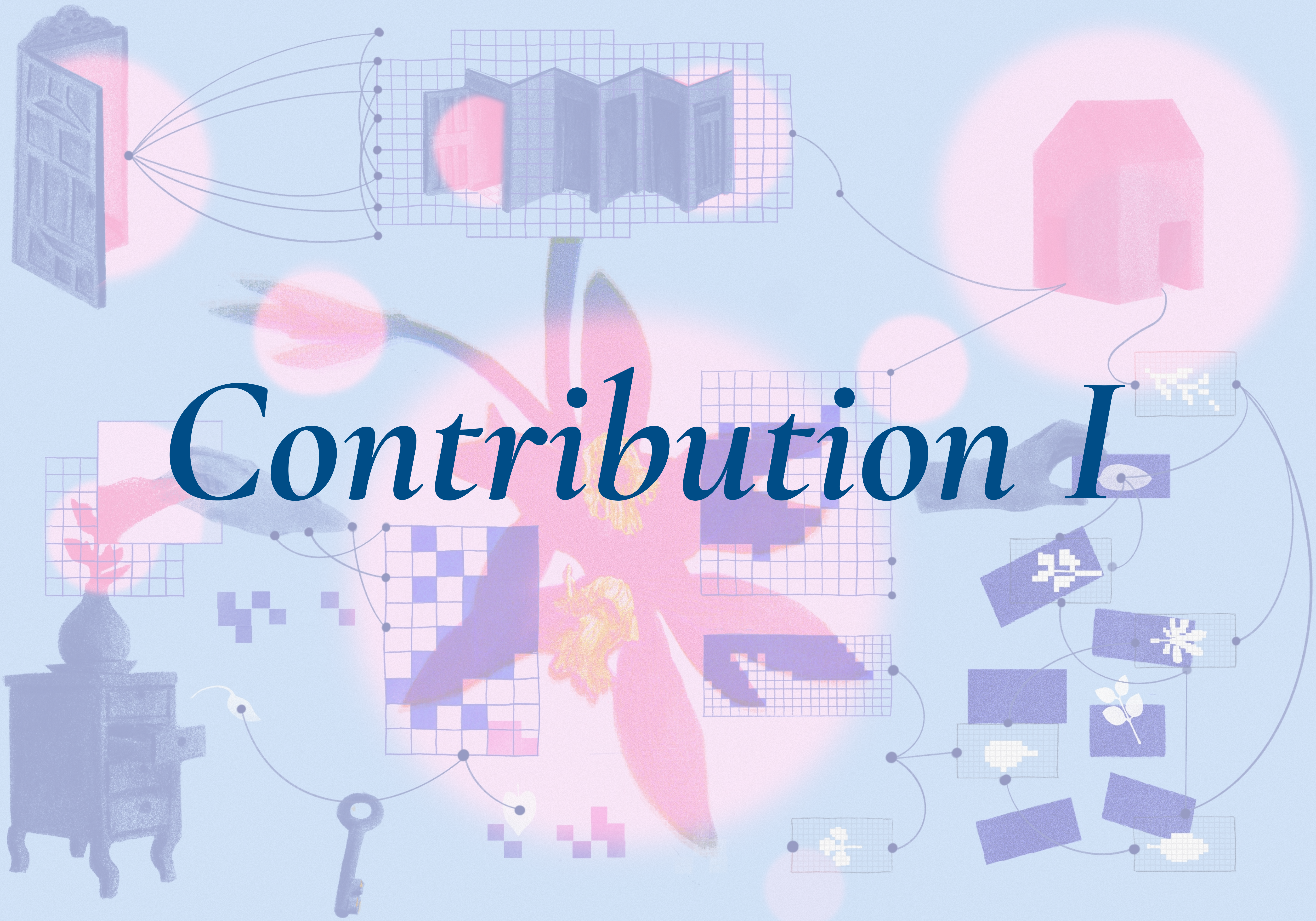
constraint violation





The multiplier accumulates the sequence of observed constraint violations.

Contribution I



Controlled Sparsity via Constrained Optimization

How I Learned to Stop Tuning Penalties & Love Constraints


Jose Gallego-Posada


Juan Ramirez


Akram Erraqabi


Yoshua Bengio


Simon Lacoste-Julien

NeurIPS 2022

(We originally wanted to write a paper on constrained optimization.
The result was a "case study" on the use of constrained optimization for training sparse neural networks.)

Sparsity via L_0 regularization

$$\min_{\tilde{\boldsymbol{\theta}},\boldsymbol{\phi}} \mathbb{E}_{z\,|\,\boldsymbol{\phi}}\left[L_{\mathscr{D}}\left(\tilde{\boldsymbol{\theta}}\odot z\right)\right]+\lambda_{\text{pen}}\mathbb{E}_{z\,|\,\boldsymbol{\phi}}\left[\|z\|_0\right]$$

L_0-"norm" penalty induces sparsity

L_0 reparametrization

Louizos et al. (2018) introduced a stochastic, differentiable reparametrization $\boldsymbol{\theta}=\tilde{\boldsymbol{\theta}}\odot z$ for training sparse neural networks

stochastic binary gates

Challenges with λ_{pen}

- Strength of the regularization is mediated by coefficient λ_{pen}.
- Tuning λ_{pen} to achieve a pre-determined sparsity level is expensive.

Louizos, C., Welling, M. and Kingma, D. *Learning Sparse Neural Networks Through L_0 Regularization*. In ICLR, 2018.

Instead of penalizing, formulate sparsity goals as L_0-norm constraints and solve the Lagrangian min-max problem

$$\min_{\tilde{\theta},\phi} \mathbb{E}_{z\,|\,\phi}\left[L_{\mathscr{D}}\left(\tilde{\theta} \odot z\right)\right] \quad \text{subject to} \quad \frac{\mathbb{E}_{z\,|\,\phi}\left[\|z\|_0\right]}{\#(\theta)} \le \epsilon$$

✔ **Interpretable hyperparameter semantics**: target sparsity level

✔ **Reliable control** over the model sparsity

Contributions

- Proposed a constrained approach for learning models with controllable levels of sparsity, highlighting its benefits with respect to the popular penalized approach
- Introduced a heuristic called ***"dual restarts"*** to avoid excessive sparsity caused by accumulation of constraint violations in the multipliers
- Through simple experimental adjustments, we **managed to successfully train sparse (Wide)ResNets** — prior experimental studies had failed at this!
- Demonstrated that **we can reliably achieve controllable sparsity levels** across many different architectures and datasets — without compromising performance

Dual Restarts

When using GDA, the multipliers can be excessively large, even at a feasible primal iterate.

Motivation of dual restarts as a "conditional" best response

The game-theoretic best response of the dual player to a primal action $(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi})$ is:

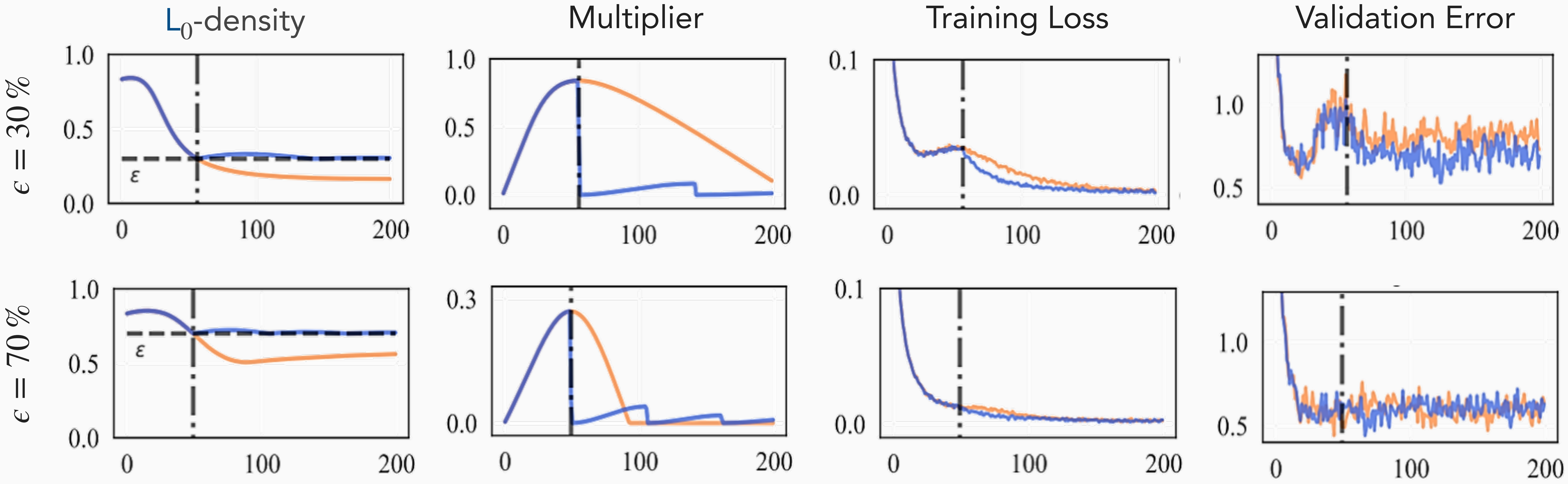
$$\lambda_{\text{CO}}^{\text{BR}}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi}) = \underset{\lambda_{\text{CO}} \geq 0}{\text{argmax}}\ \mathfrak{L}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi}, \lambda_{\text{CO}}) = \underset{\lambda_{\text{CO}} \geq 0}{\text{argmax}}\ f(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi}) + \lambda_{\text{CO}}^{\top} \left(g(\boldsymbol{\phi}) - \epsilon \right)$$

This is a linear program whose solution depends purely on the feasibility of $(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi})$:

$$\lambda_{\text{CO}}^{\text{BR}}(\tilde{\boldsymbol{\theta}}, \boldsymbol{\phi}) = \begin{cases} \infty & \text{if} \quad g(\boldsymbol{\phi}) > \epsilon \\ \mathbb{R}^{+} & \text{if} \quad g(\boldsymbol{\phi}) = \epsilon \\ 0 & \text{if} \quad g(\boldsymbol{\phi}) < \epsilon \end{cases}$$

Training Dynamics

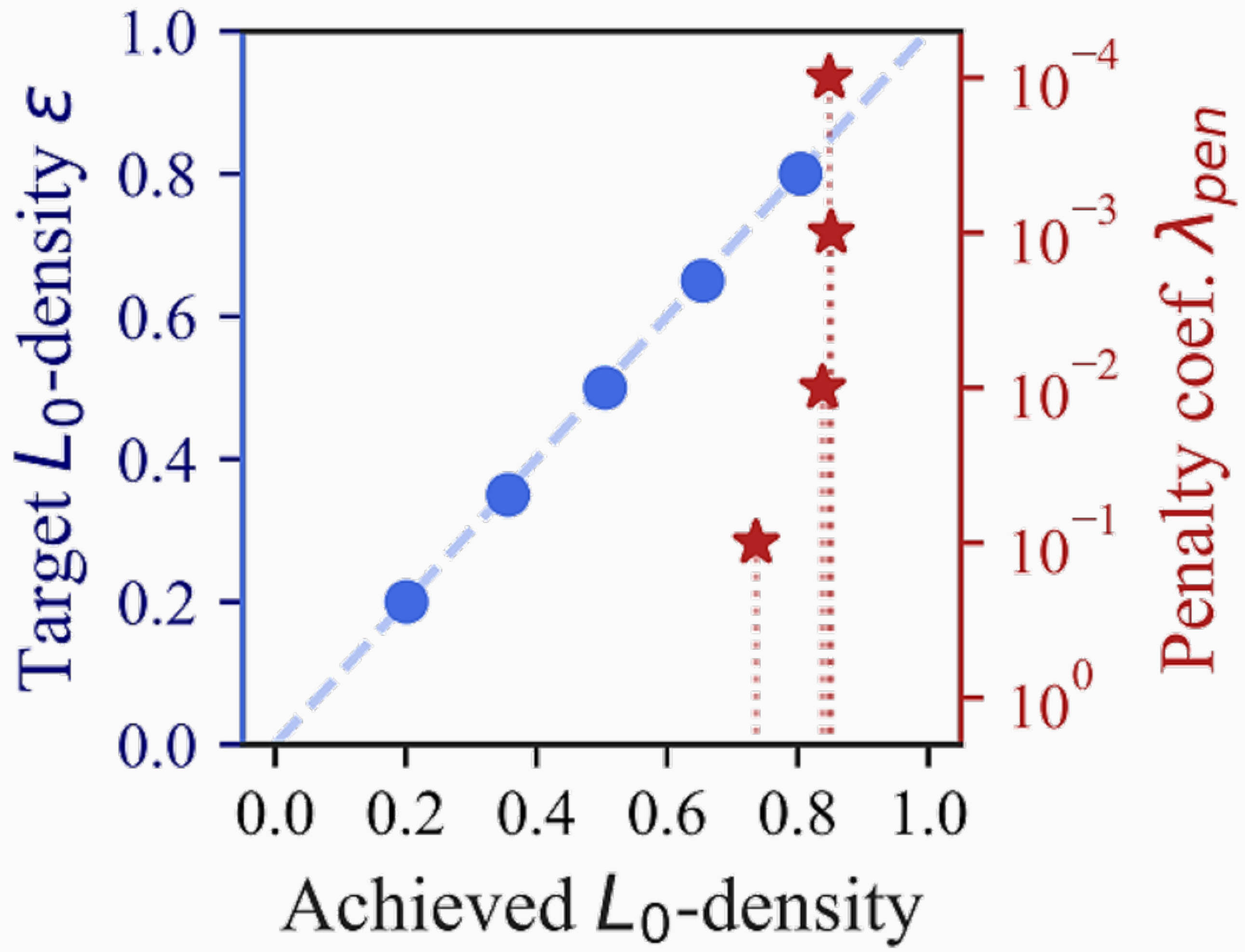
Dataset: MNIST Model: LeNet5

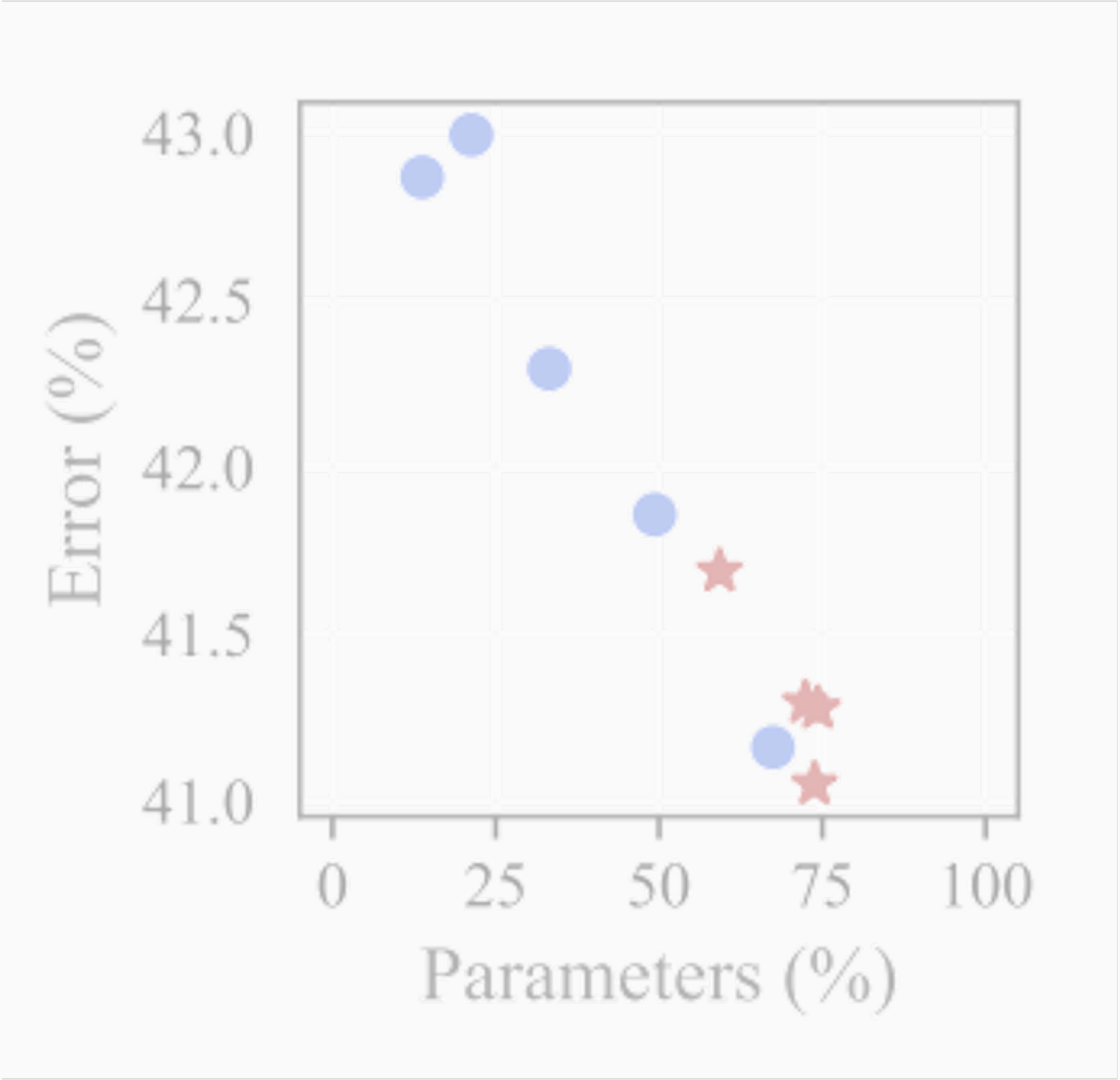


Dual Restarts No Dual Restarts First Feasibility

Achieving controlled sparsity

Dataset: Tiny-ImageNet   Model: ResNet18

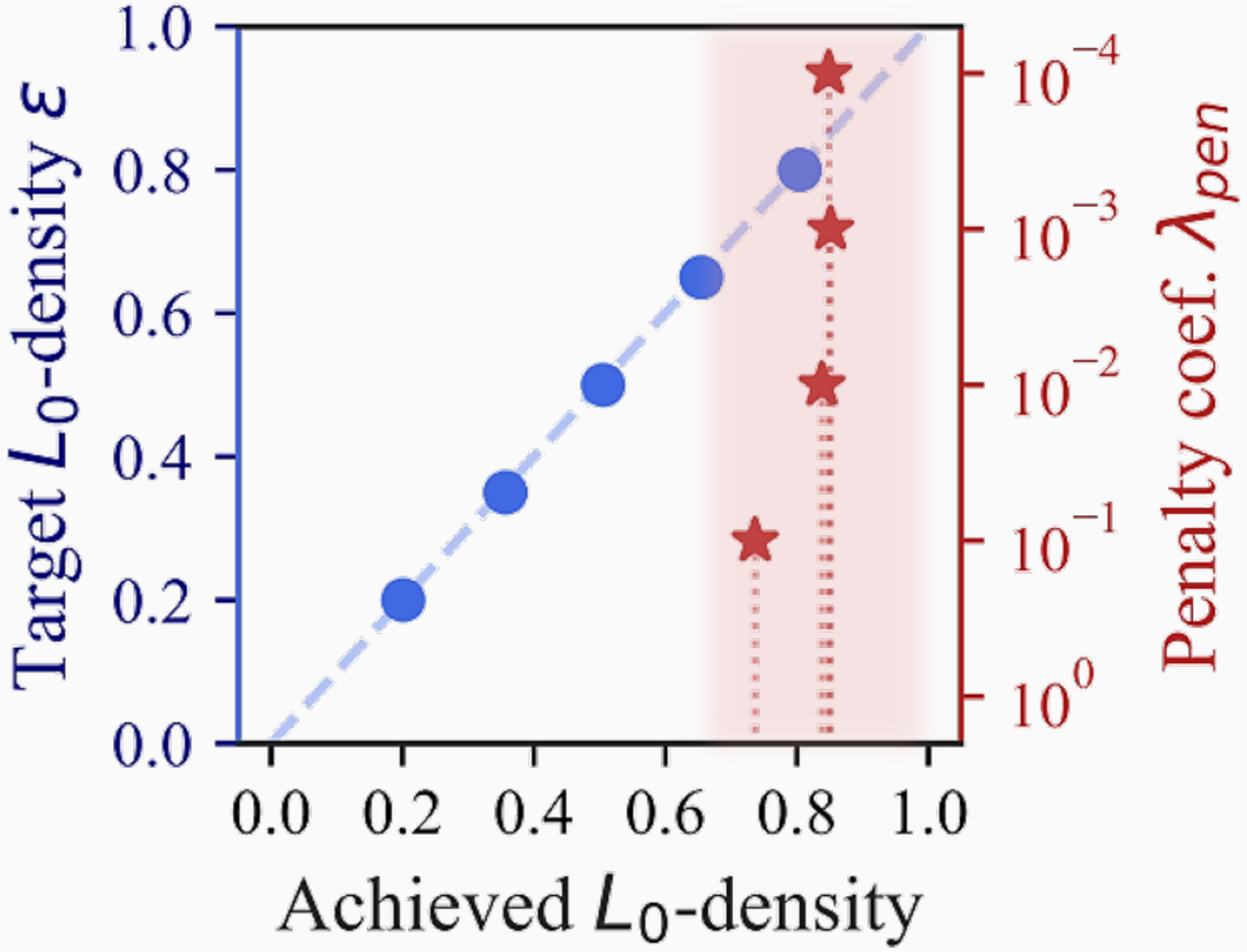


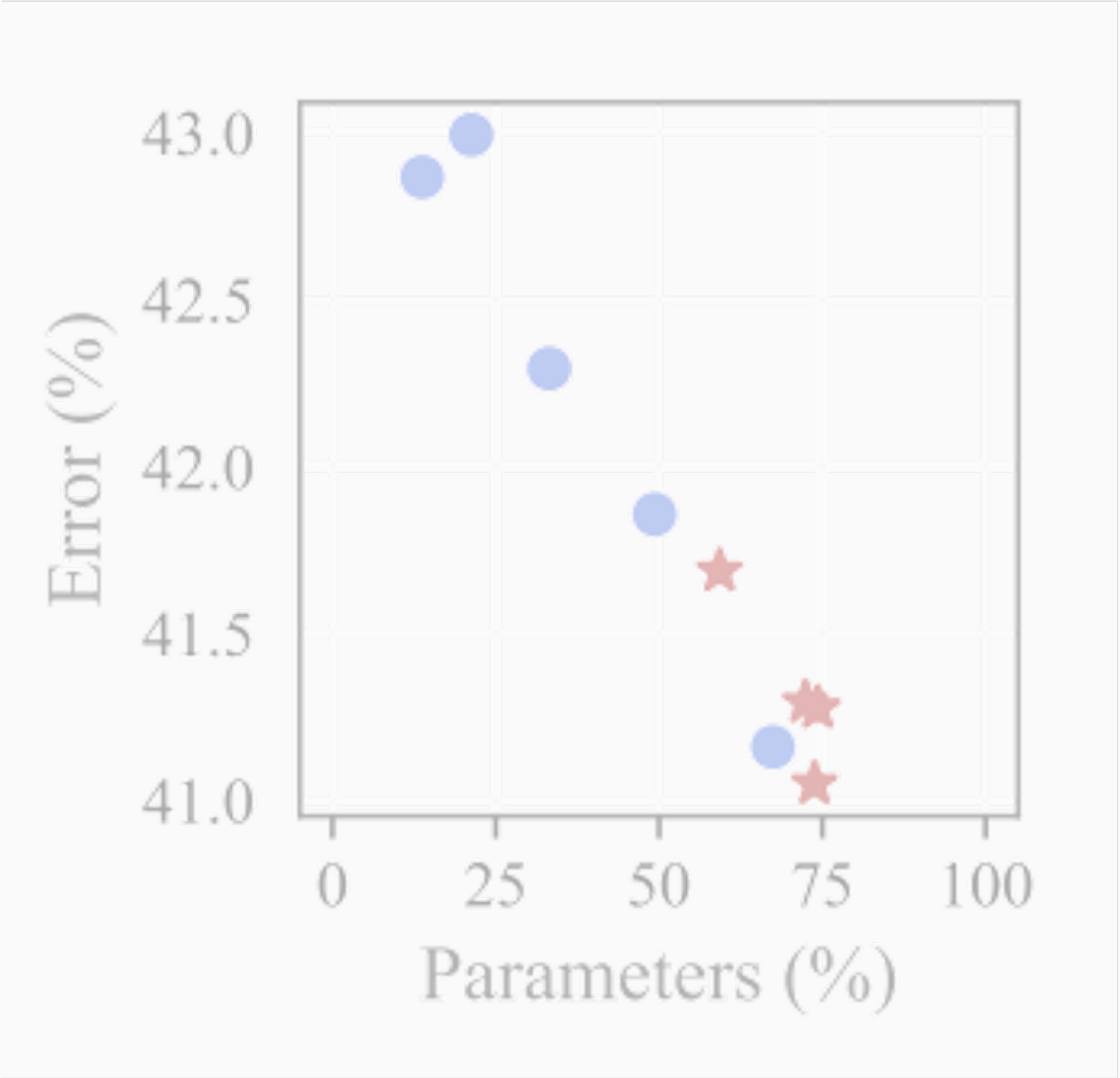


● Constrained ★ Penalized

Achieving controlled sparsity

Dataset: Tiny-ImageNet   Model: ResNet18

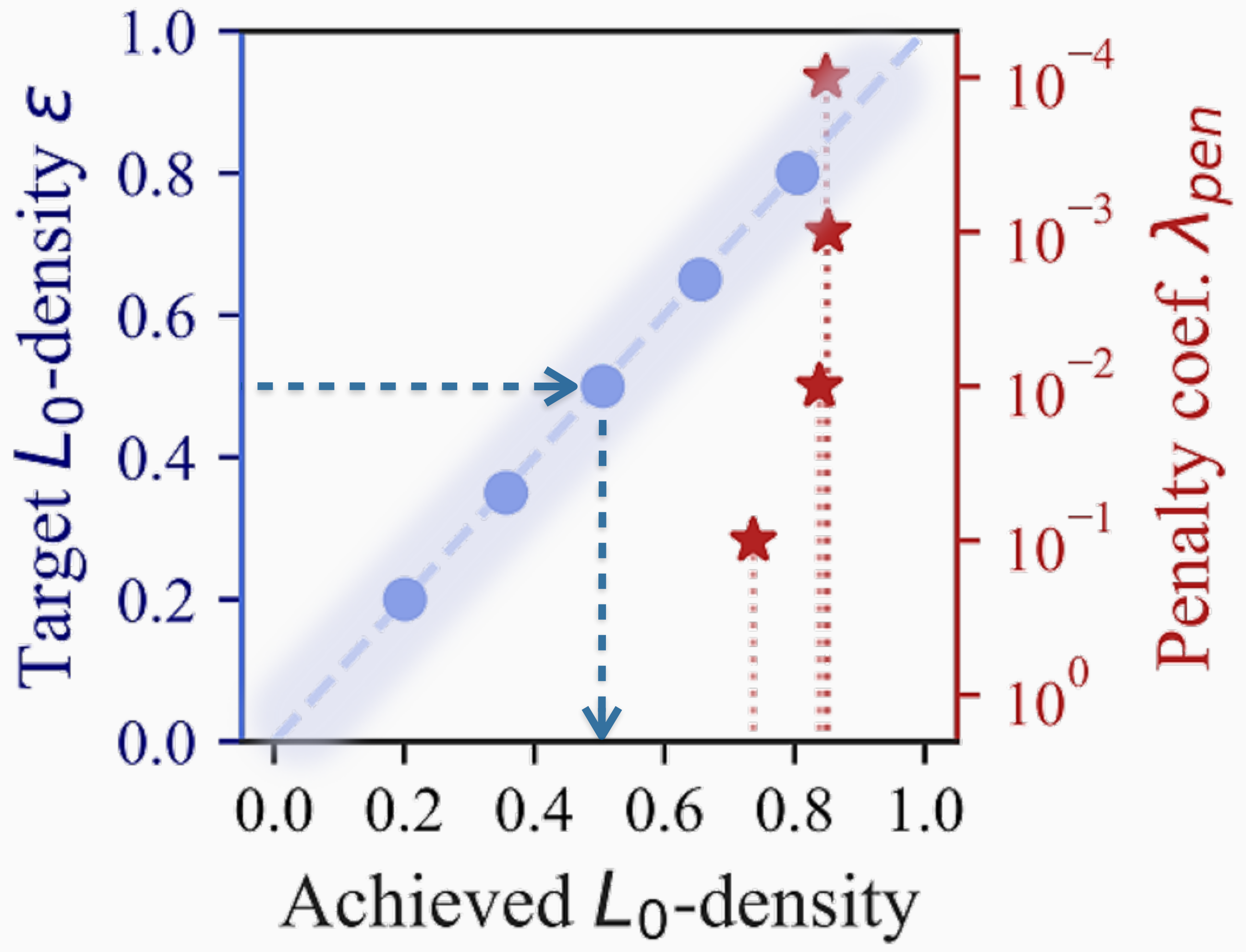


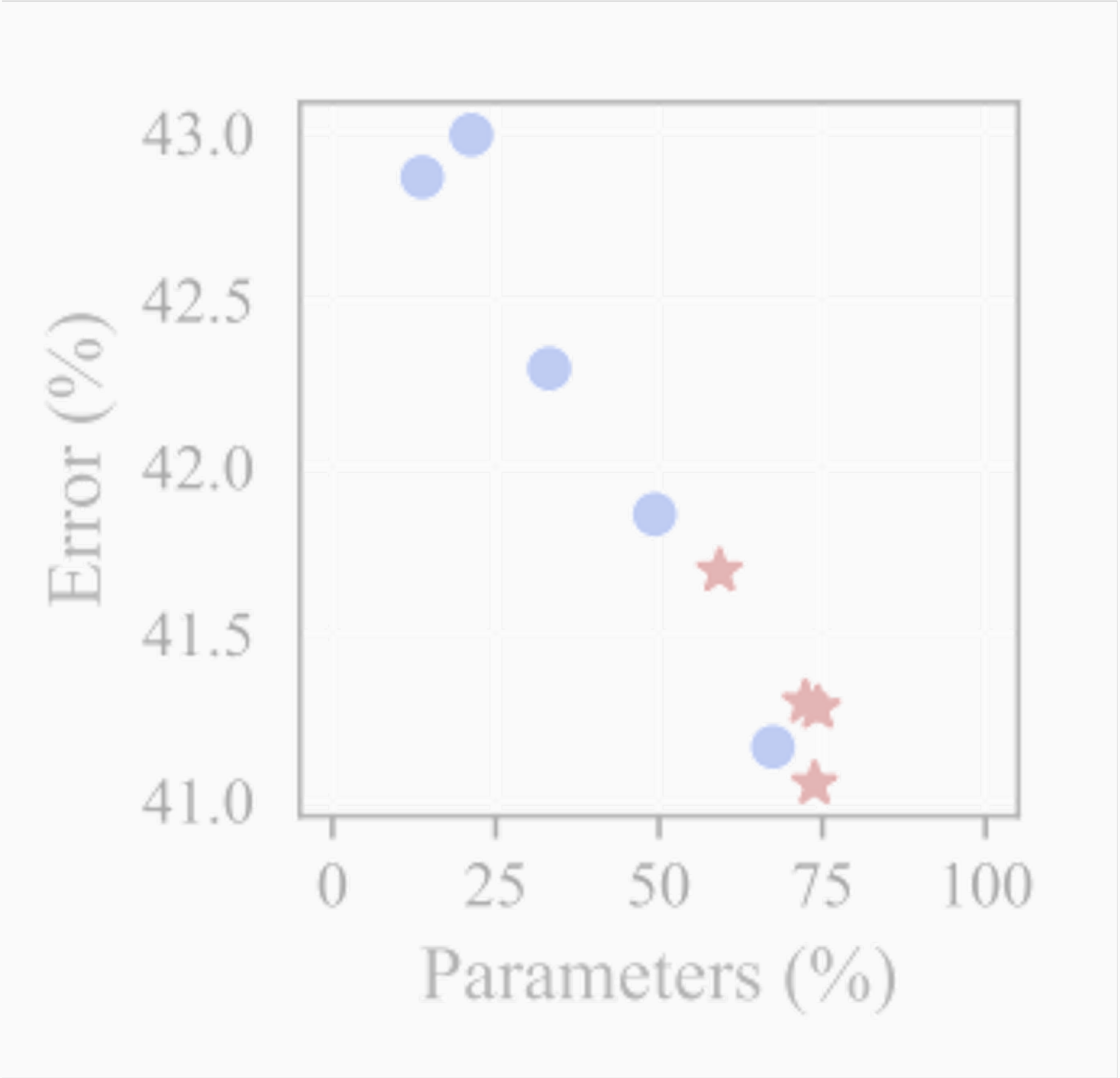


● Constrained ★ Penalized

Achieving controlled sparsity

Dataset: Tiny-ImageNet Model: ResNet18

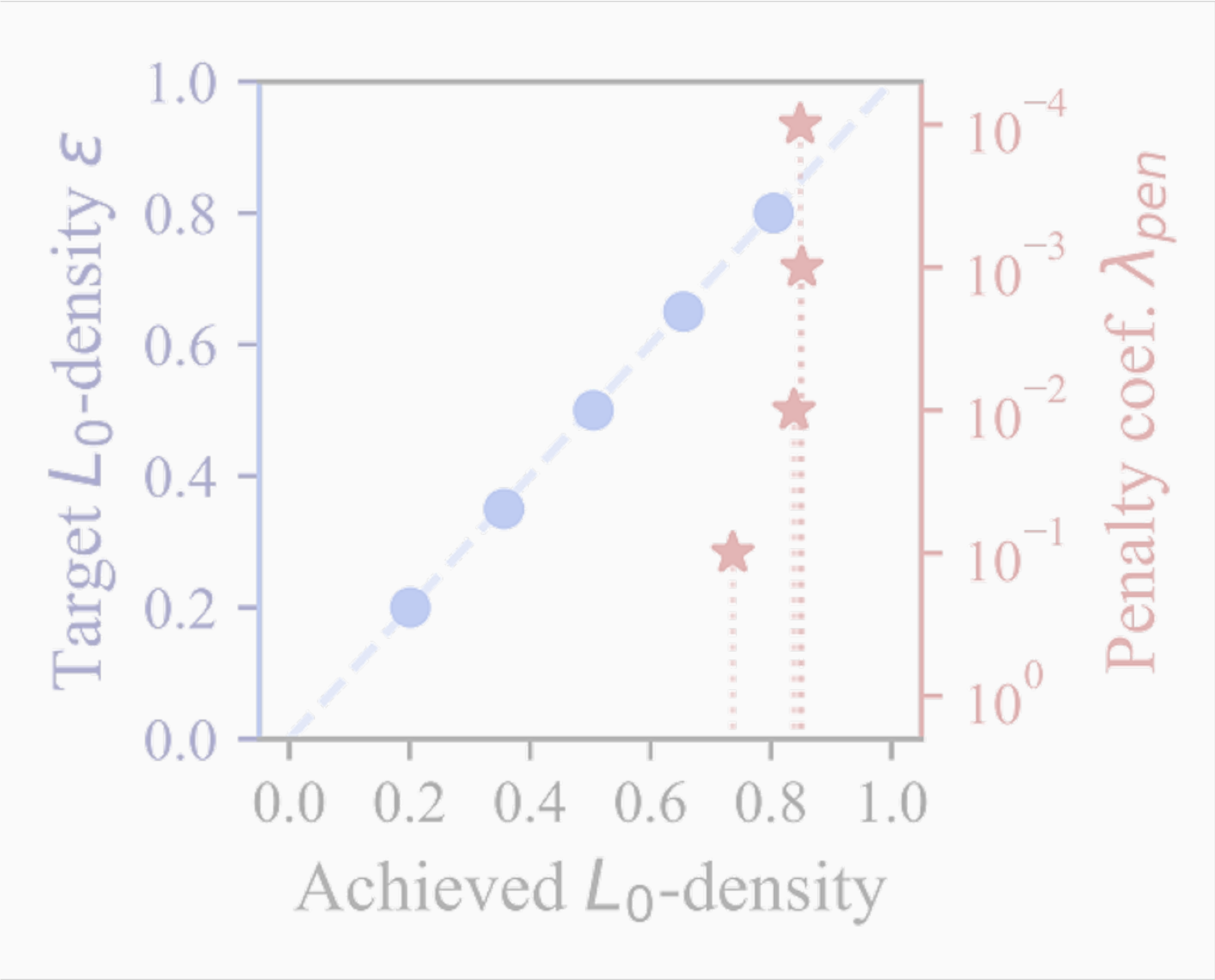


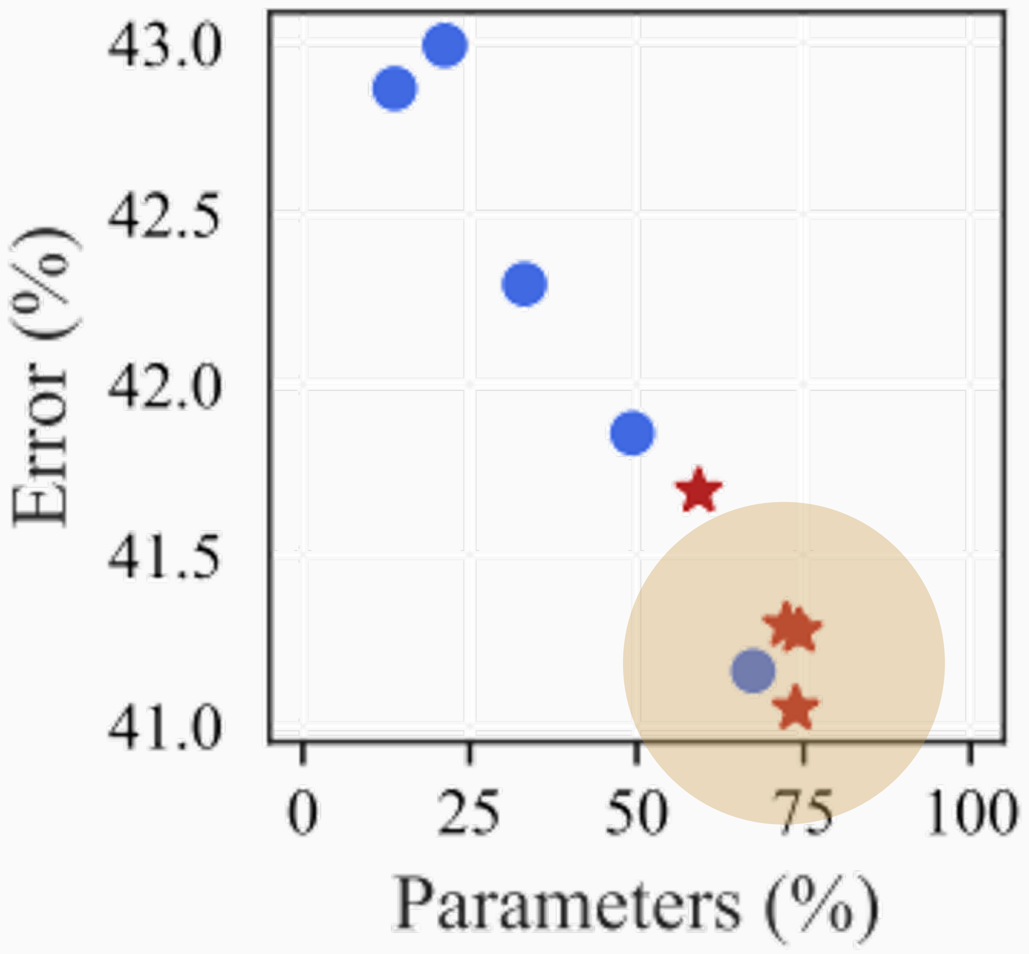


● Constrained ★ Penalized

… while retaining performance!

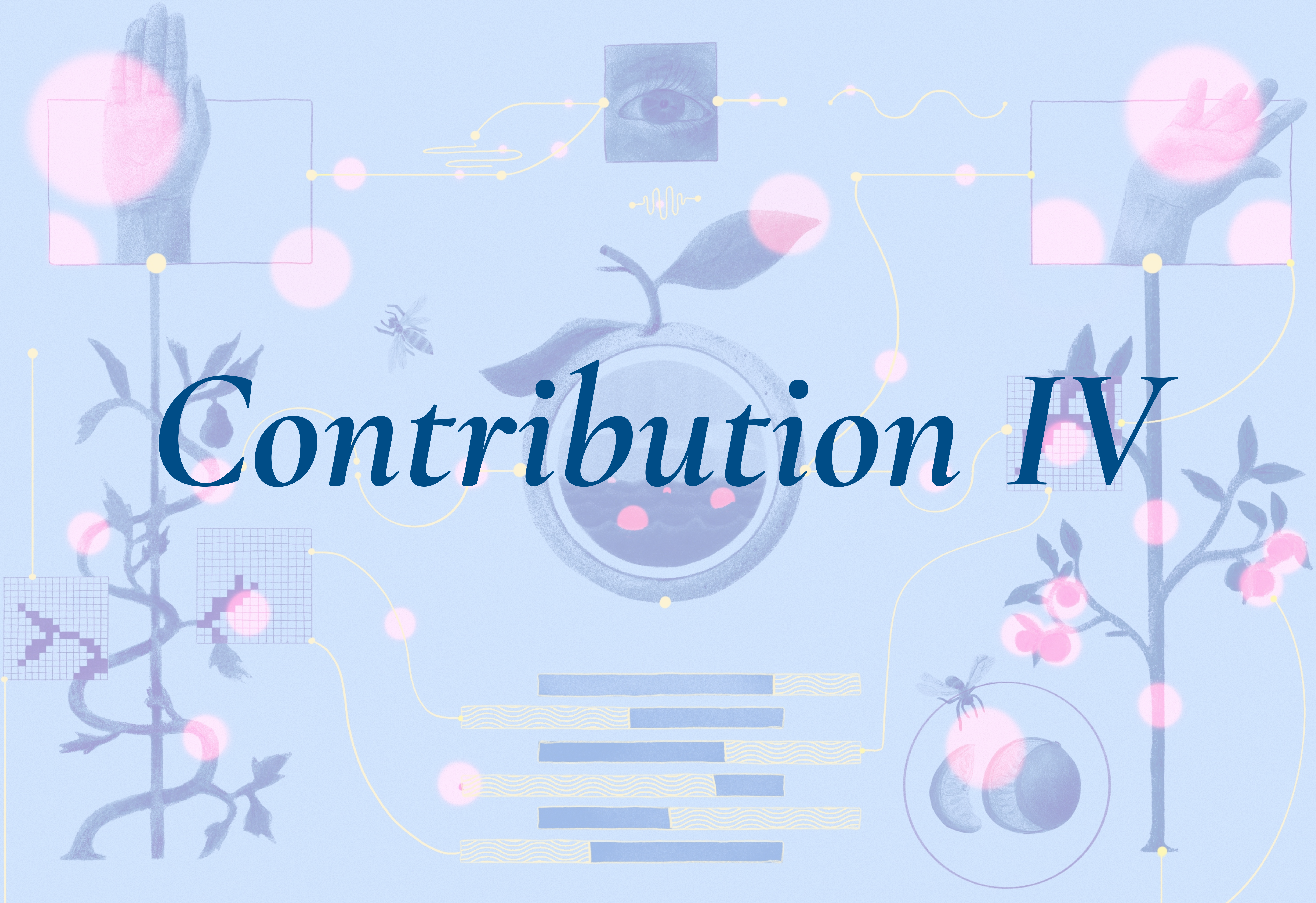
Dataset: Tiny-ImageNet   Model: ResNet18





● Constrained ★ Penalized

Contribution IV



On PI controllers for updating Lagrange multipliers in constrained optimization



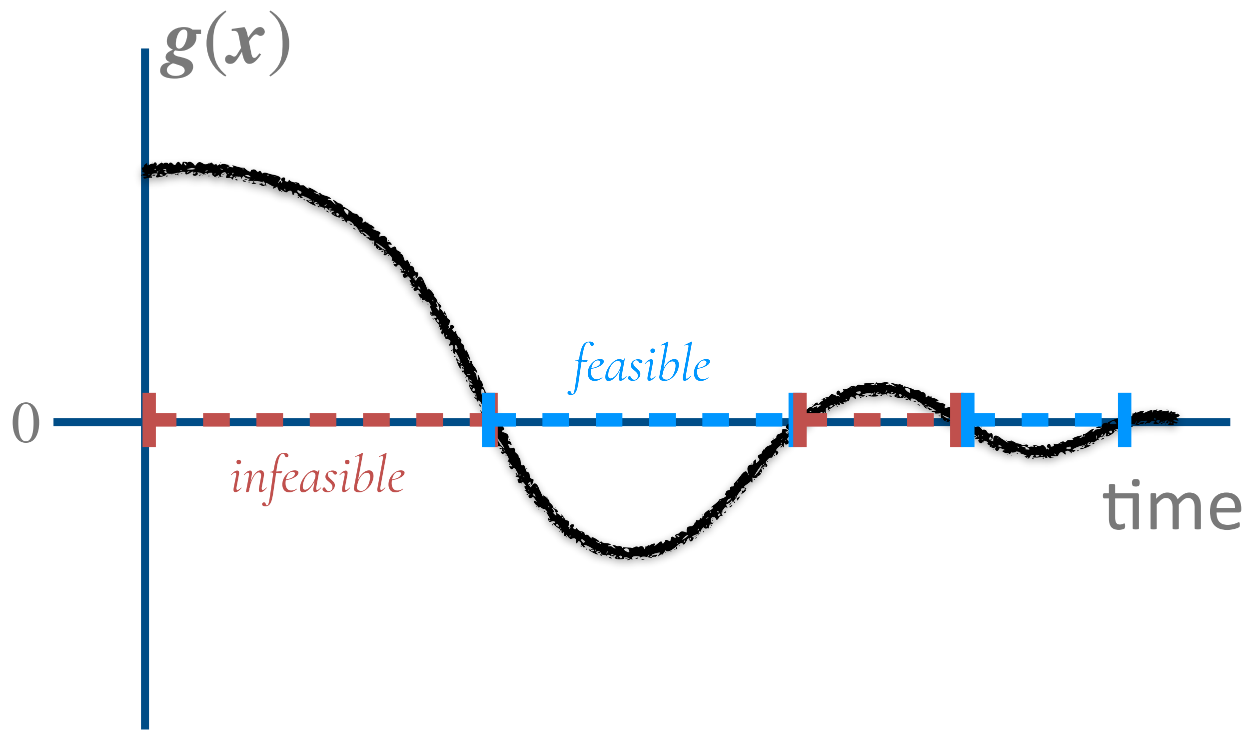


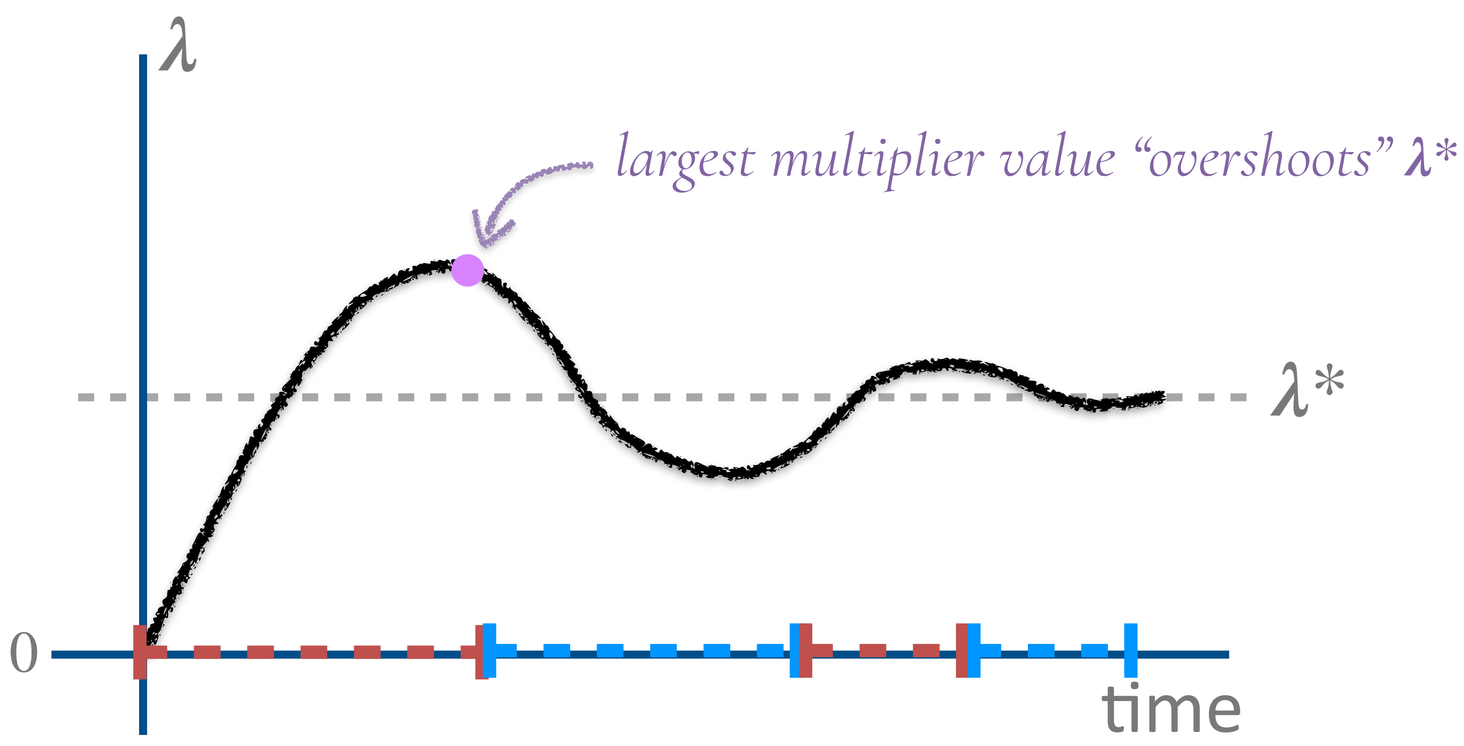
Motahareh Sohrabi
Juan Ramirez
Tianyue H. Zhang
Simon Lacoste-Julien
Jose Gallego-Posada

ICML 2024

Dynamics of GDA

$$\lambda_{k+1} = \left[\lambda_k + \eta_{\text{dual}} \nabla_\lambda \mathfrak{L}(x_k, \lambda_k, \mu_k)\right]^+ = \left[\lambda_k + \eta_{\text{dual}}\, g(x_k)\right]^+$$





The multiplier accumulates/*integrates* the sequence of observed constraint violations

What we are looking for

Shortcomings of GDA

- GDA may result in overshoot and oscillations (Gidel et al. 2019; Stooke at al. 2020)
- Especially problematic in safety-related applications

Goal and scope

- **Reliable and robust** approach for solving Lagrangian optimization problems
- That **does not modify** training "recipe" for primal variables

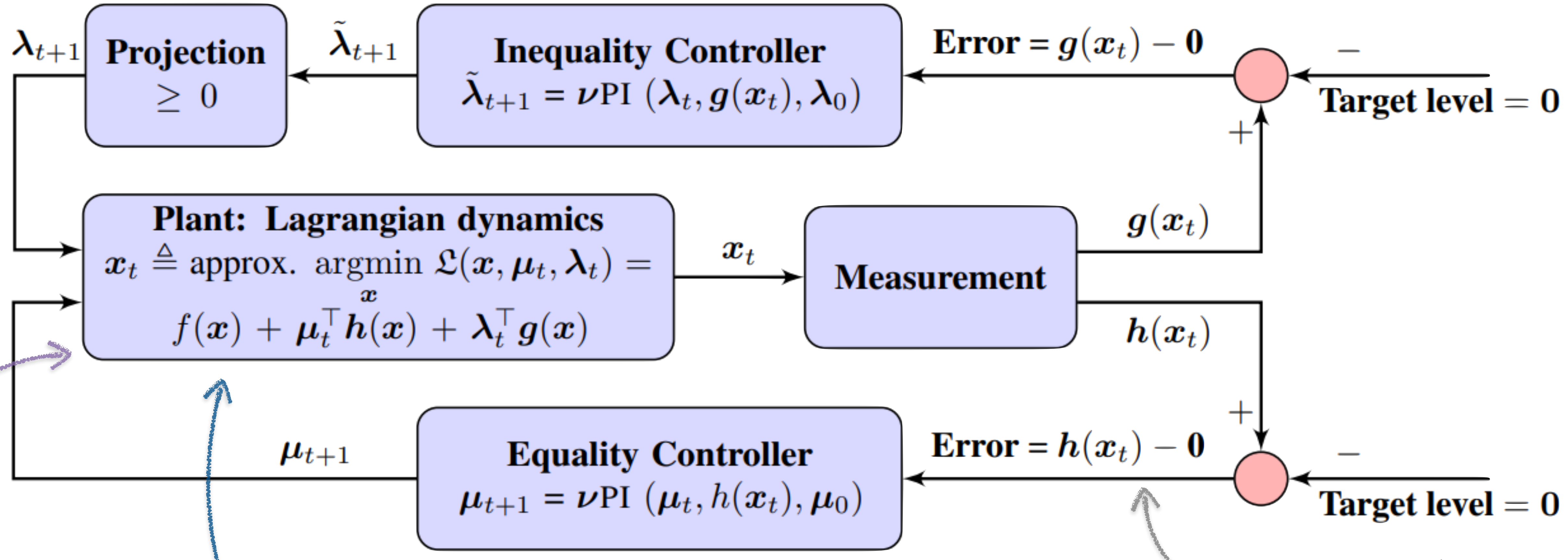
Achieving this goal enables wider adoption of Lagrangian optimization in deep learning!

Gidel, G., Askari, R., Pezeshki, M., LePriol, R., Huang, G., Lacoste-Julien, S., and Mitliagkas, I. *Negative Momentum for Improved Game Dynamics.* In AISTATS, 2019.
Stooke, A., Achiam, J., and Abbeel, P. *Responsive Safety in Reinforcement Learning by PID Lagrangian Methods.* In ICML, 2020.

Dynamical system's view of CO

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{subject to } \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0} \text{ and } \boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}$$

ensure non-negativity of inequality multipliers



plug-in your favourite primal optimizer choice

constraint violation is the error signal for the controller

multipliers "tilt" the primal gradient

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla \boldsymbol{g}_i(\boldsymbol{x}^*) + \sum_{i=1}^{n} \boldsymbol{\mu}_i^* \nabla \boldsymbol{h}_i(\boldsymbol{x}^*)$$

νPI control for constrained optimization

Algorithm: νPI update on parameter $\boldsymbol{\theta}$

Args: EMA coefficient ν, proportional (κ_p) and integral (κ_i) gains; initial conditions $\boldsymbol{\xi}_0$ and $\boldsymbol{\theta}_0$

1. Measure the current system error $\boldsymbol{e}_t$
2. $\boldsymbol{\xi}_t \leftarrow \nu \boldsymbol{\xi}_{t-1} + (1-\nu)\boldsymbol{e}_t$ (for $t \geq 1$)
3. $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_0 + \kappa_p \boldsymbol{\xi}_t + \kappa_i \sum_{\tau=0}^{t} \boldsymbol{e}_\tau$

Recursively, $\boldsymbol{\theta}_1 \leftarrow \boldsymbol{\theta}_0 + \kappa_p \boldsymbol{\xi}_0 + \kappa_i \boldsymbol{e}_0$

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \kappa_i \boldsymbol{e}_t + \kappa_p \left(\boldsymbol{\xi}_t - \boldsymbol{\xi}_{t-1}\right)$$

General case

like ∇-ascent

new term looks at ***change*** *in constraint satisfaction!*

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \kappa_i \boldsymbol{e}_t + \kappa_p \left(\boldsymbol{e}_t - \boldsymbol{e}_{t-1}\right)$$

Case $\nu = 0$

νPI generalizes momentum methods

Theorem

Polyak $\gamma = 0$; Nesterov $\gamma = 1$

Under the same initialization $\boldsymbol{\theta}_0$, UnifiedMomentum(α, $\beta \neq 1$, γ) is a special case of the νPI algorithm with the hyperparameter choices:

$$\nu \leftarrow \beta \qquad \boldsymbol{\xi}_0 \leftarrow (1-\beta)\boldsymbol{e}_0$$

$$\kappa_i \leftarrow \frac{\alpha}{1-\beta} \qquad \kappa_p \leftarrow -\frac{\alpha\beta}{(1-\beta)^2}[1-\gamma(1-\beta)]$$

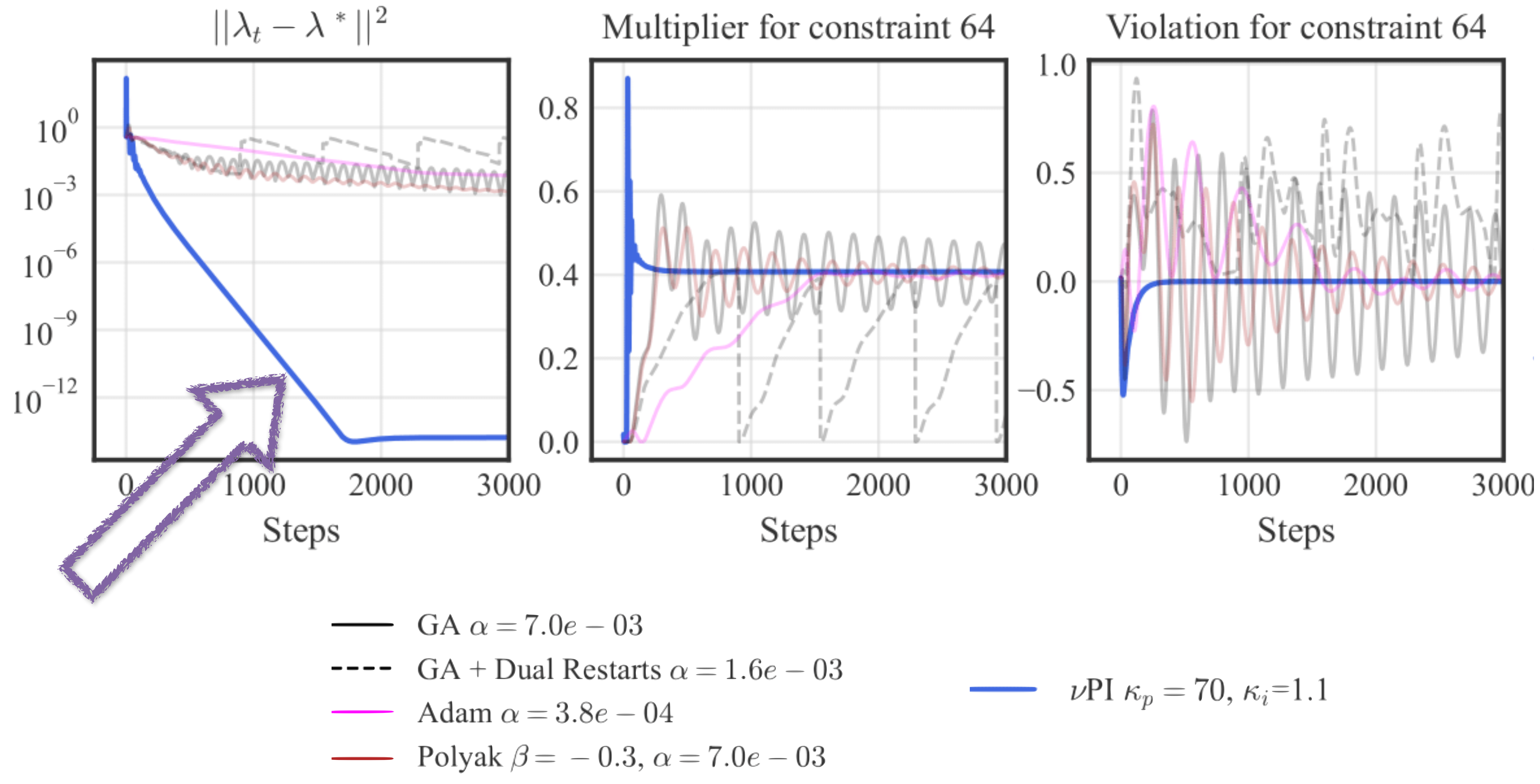
Shen, L., Chen, C., Zou, F., Jie, Z., Sun, J., and Liu, W. A. *Unified Analysis of AdaGrad with Weighted Aggregation and Momentum Acceleration.* In IEEE TNNLS, 2018.

νPI generalizes momentum methods

Algorithm	$\boldsymbol{\xi}_0$	κ_p	κ_i	ν
UNIFIEDMOMENTUM(α, β, γ)	$(1-\beta)\boldsymbol{e}_0$	$-\frac{\alpha\beta}{(1-\beta)^2}[1-\gamma(1-\beta)]$	$\frac{\alpha}{1-\beta}$	β
POLYAK(α, β)	$(1-\beta)\boldsymbol{e}_0$	$-\frac{\alpha\beta}{(1-\beta)^2}$	$\frac{\alpha}{1-\beta}$	β
NESTEROV(α, β)	$(1-\beta)\boldsymbol{e}_0$	$-\frac{\alpha\beta^2}{(1-\beta)^2}$	$\frac{\alpha}{1-\beta}$	β
PI	$\boldsymbol{e}_0$	κ_p	κ_i	0
OPTIMISTICGRADIENTASCENT(α)	$\boldsymbol{e}_0$	α	α	0
$\boldsymbol{\nu}$PI $(\kappa_i, \kappa_p, \nu)$ in practice	$\mathbf{0}$	κ_i	κ_p	ν
GRADIENTASCENT(α)	–	0	α	0

SVM task

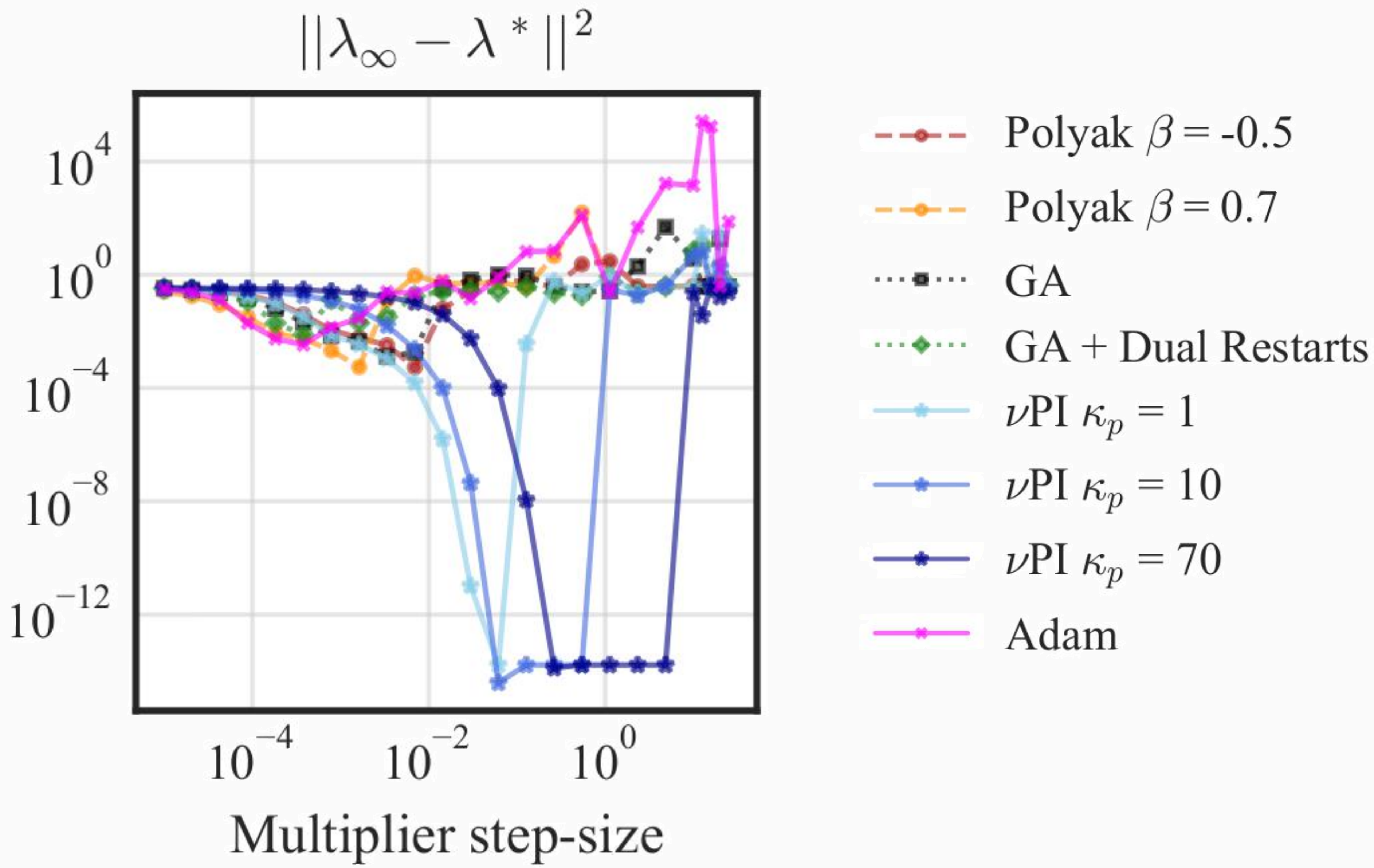
Of all attempted optimizers*, **only νPI converged successfully to the true solution!**

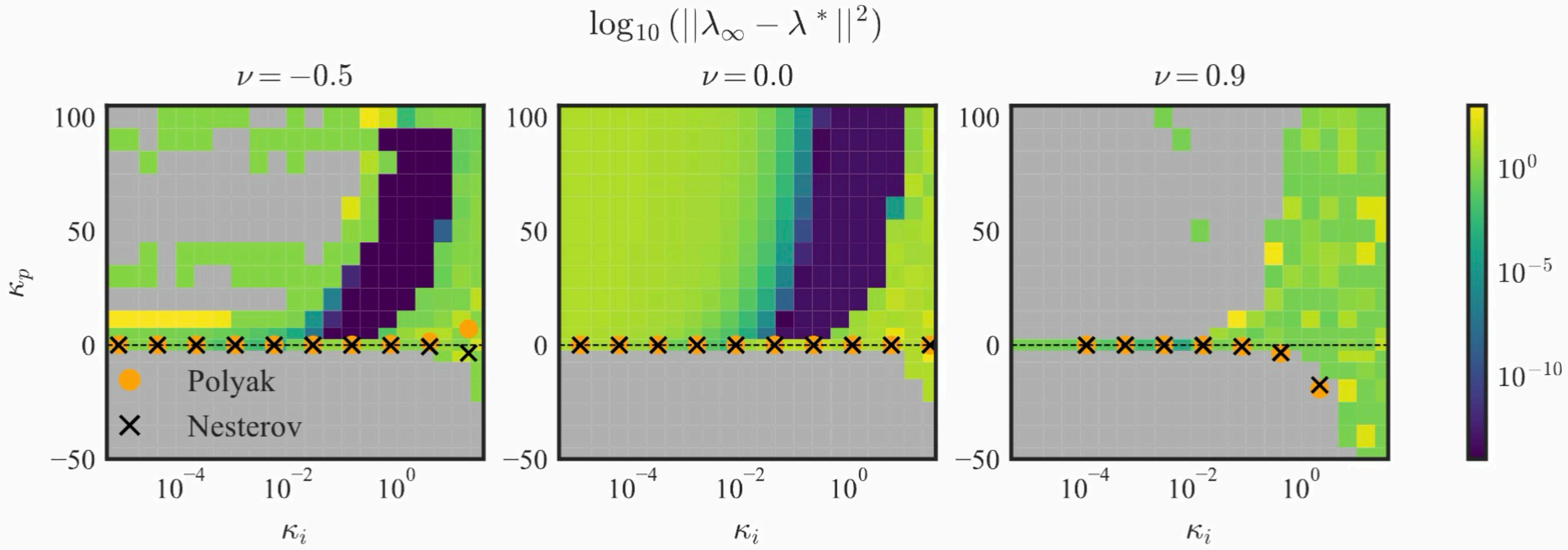


Showing best hyperparameters for each optimizer after grid-search aiming to minimize the distance to λ after 5.000 iterations

Robustness

Higher values of κ_p allow for choosing **larger values of κ_i** (multiplier step-size) and **over a wider range**, while still achieving convergence.



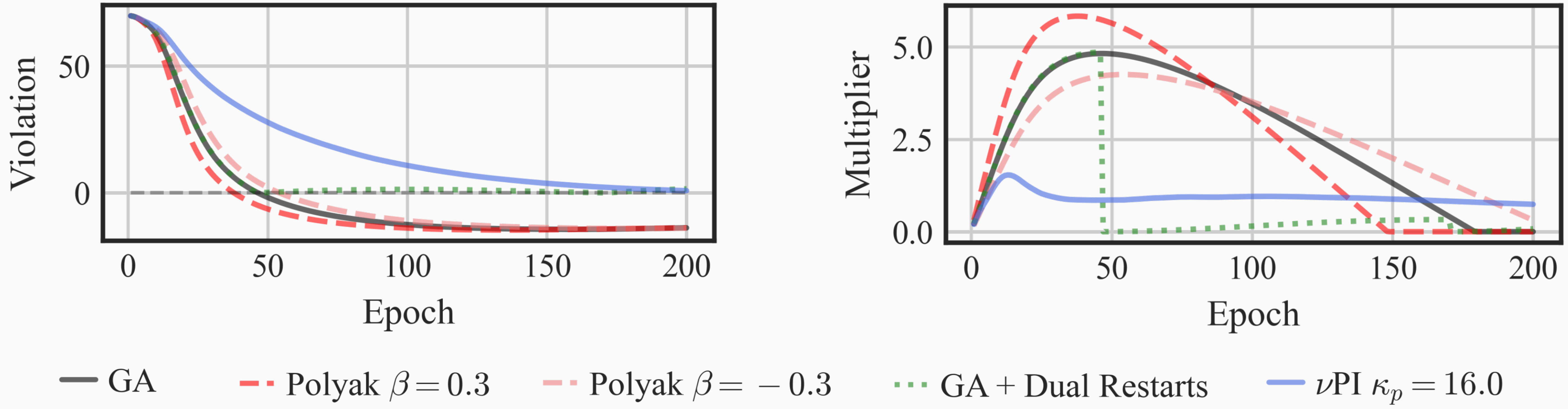


νPI provides additional flexibility compared to Polyak and Nesterov which is crucial for achieving convergence in this task.

Revisiting L_0-constrained problem

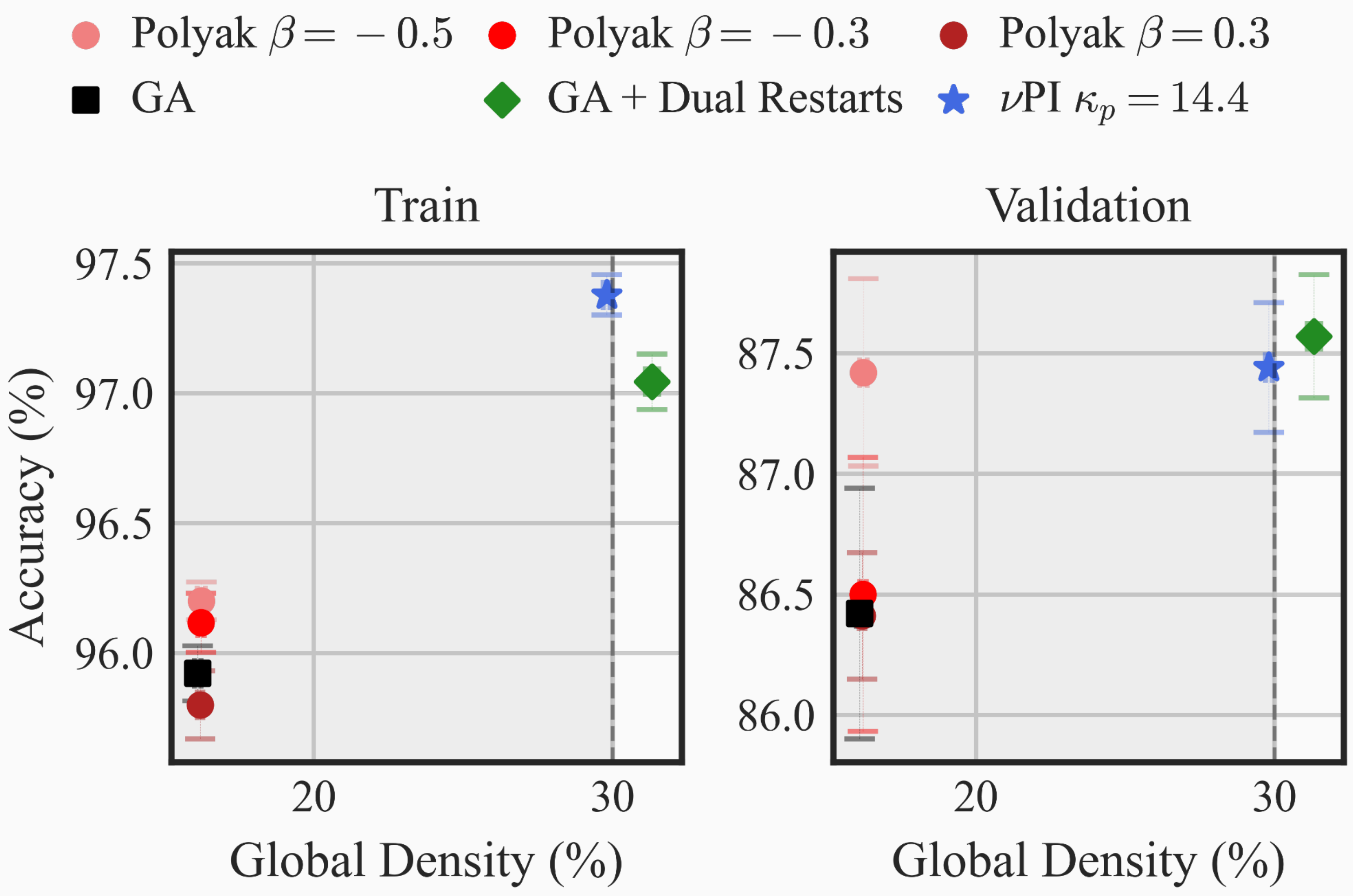
$$\min_{\tilde{\theta},\phi} \mathbb{E}_{z\,|\,\phi}\left[L_{\mathscr{D}}\left(\tilde{\theta}\odot z\right)\right] \quad \text{subject to} \quad \frac{\mathbb{E}_{z\,|\,\phi}\left[\|z\|_0\right]}{\#(\theta)} \leq \epsilon$$

νPI delivers **stable multiplier dynamics without constraint overshoot**

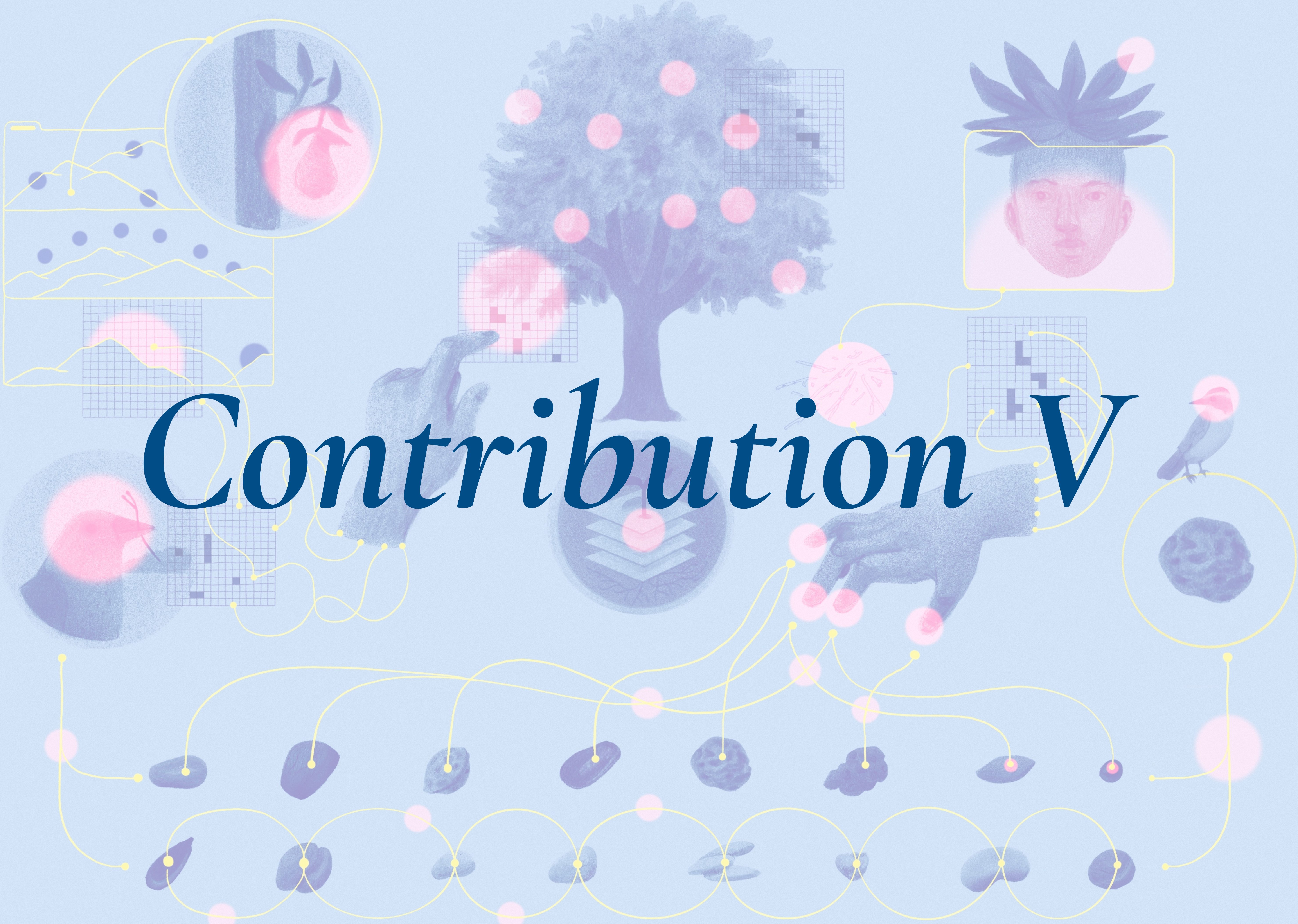


Impact on performance

νPI achieves high accuracy and tightly respects the constraints, without overshooting



Contribution V



Cooper:
A Library for Constrained Optimization in Deep Learning

Jose Gallego-Posada

Juan Ramirez

Meraj Hashemizadeh

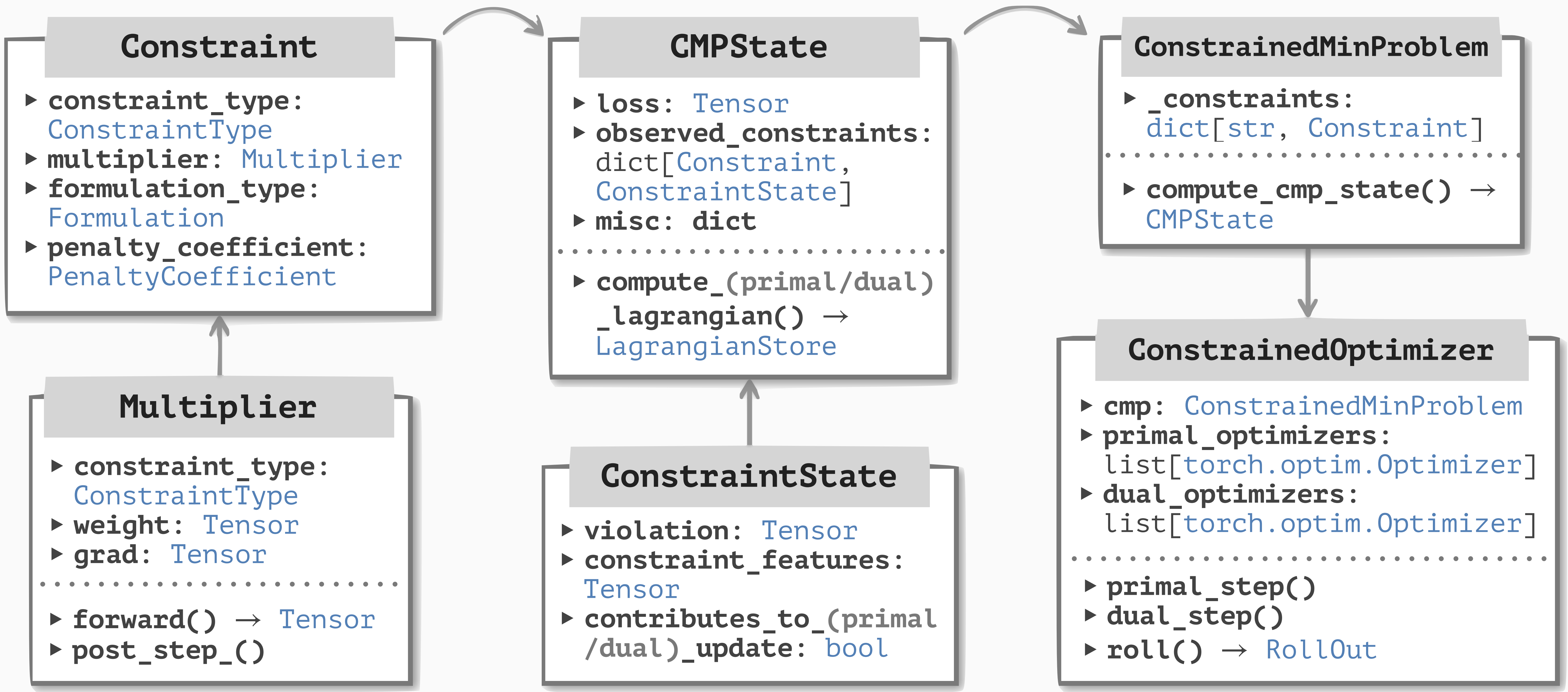
Simon Lacoste-Julien

JMLR MLOSS 2024 *(under submission)*

Cooper

a general-purpose, deep learning-first library for constrained optimization, built on PyTorch.

Cooper's class overview



Cotter et al. *TensorFlow Constrained Optimization (TFCO).*

Conclusions & Perspectives

Constrained optimization is an up-and-coming research direction

- As ML becomes a "technology", ensuring compliance with government regulations and industry standards is crucial next-step
- Constrained optimization is a rich field, ripe for integration by ML community
- Socially impactful research; inter-disciplinary relevance

Main challenges when solving constrained problems in machine learning

- Optimization dynamics
- Non-differentiable constraints (“*proxy-constraints*” from Cotter et al. (2019))
- Generalization properties for loss **and** constraints
- Feasibility: always? if not, how fast?

(Some) Open questions

- How to deal with constraints that are difficult to quantify?
- Why do GDA-like schemes work in practical Lagrangian problems?
- What is the role of overparametrization in constrained optimization?
- How can we improve the Lagrange multipliers further?
- How can we make constrained techniques usable "during inference"?
- What is next for Cooper?

Thank you!