Contributions to stochastic bilevel optimization

Mathieu DAGRÉOU

Under the supervision of Pierre Ablin, Thomas Moreau and Samuel Vaiter

Ph.D. Commitee:	Aurélien Bellet	*Inria Montpellier*
	Peter Ochs	*Saarland University*
	Émilie Chouzenoux	*Inria Saclay*
	Julien Mairal	*Inria Grenoble*
	Édouard Pauwels	*Toulouse School of Economics*

Bilevel optimization in machine learning

Bilevel optimization in machine learning

- Mathematical formalism where the problem we want to solve depends on the solution of another problem

Bilevel optimization in machine learning

- Mathematical formalism where the problem we want to solve depends on the solution of another problem

- Attention in the ML community because of its ability to model many situations

Bilevel optimization in machine learning

Model Selection via Bilevel Optimization

Kristin P. Bennett, Jing Hu, Xiaoyun Ji, Gautam Kunapuli, and Jong-Shi Pang

- Mathematical formalism where the problem we want to solve depends on the solution of another problem

- Attention in the ML community because of its ability to model many situations

Bilevel optimization in machine learning

- Mathematical formalism where the problem we want to solve depends on the solution of another problem

- Attention in the ML community because of its ability to model many situations

Model Selection via Bilevel Optimization

Kristin P. Bennett, Jing Hu, Xiaoyun Ji, Gautam Kunapuli, and Jong-Shi Pang

DARTS: DIFFERENTIABLE ARCHITECTURE SEARCH

Hanxiao Liu*	**Karen Simonyan**	**Yiming Yang**
CMU	DeepMind	CMU
hanxiaol@cs.cmu.com	simonyan@google.com	yiming@cs.cmu.edu

Bilevel optimization in machine learning

- Mathematical formalism where the problem we want to solve depends on the solution of another problem

- Attention in the ML community because of its ability to model many situations

Model Selection via Bilevel Optimization

Kristin P. Bennett, Jing Hu, Xiaoyun Ji, Gautam Kunapuli, and Jong-Shi Pang

DARTS: DIFFERENTIABLE ARCHITECTURE SEARCH

Hanxiao Liu*
CMU
hanxiaol@cs.cmu.com

Karen Simonyan
DeepMind
simonyan@google.com

Yiming Yang
CMU
yiming@cs.cmu.edu

Bilevel Optimization to Learn Training Distributions for Language Modeling under Domain Shift

David Grangier, Pierre Ablin, Awni Hannun
Apple, {grangier,pablin,awni}@apple.com

Bilevel optimization in machine learning

- Mathematical formalism where the problem we want to solve depends on the solution of another problem

- Attention in the ML community because of its ability to model many situations

Model Selection via Bilevel Optimization

Kristin P. Bennett, Jing Hu, Xiaoyun Ji, Gautam Kunapuli, and Jong-Shi Pang

DARTS: DIFFERENTIABLE ARCHITECTURE SEARCH

Hanxiao Liu*
CMU
hanxiaol@cs.cmu.com

Karen Simonyan
DeepMind
simonyan@google.com

Yiming Yang
CMU
yiming@cs.cmu.edu

Bilevel Optimization to Learn Training Distributions for Language Modeling under Domain Shift

David Grangier, Pierre Ablin, Awni Hannun
Apple, {grangier,pablin,awni}@apple.com

DADA: Differentiable Automatic Data Augmentation

Yonggang Li*[1], Guosheng Hu*[2,3], Yongtao Wang†[1], Timothy Hospedales[4], Neil M. Robertson[2,3], and Yongxin Yang[4]

Bilevel optimization in machine learning

- Mathematical formalism where the problem we want to solve depends on the solution of another problem

- Attention in the ML community because of its ability to model many situations

Model Selection via Bilevel Optimization

Kristin P. Bennett, Jing Hu, Xiaoyun Ji, Gautam Kunapuli, and Jong-Shi Pang

DARTS: DIFFERENTIABLE ARCHITECTURE SEARCH

Hanxiao Liu*
CMU
hanxiaol@cs.cmu.com

Karen Simonyan
DeepMind
simonyan@google.com

Yiming Yang
CMU
yiming@cs.cmu.edu

Bilevel Optimization to Learn Training Distributions for Language Modeling under Domain Shift

David Grangier, Pierre Ablin, Awni Hannun
Apple, {grangier,pablin,awni}@apple.com

DADA: Differentiable Automatic Data Augmentation

Yonggang Li[*1], Guosheng Hu[*2,3], Yongtao Wang[†1], Timothy Hospedales[4], Neil M. Robertson[2,3], and Yongxin Yang[4]

Deep Equilibrium Models

Shaojie Bai
Carnegie Mellon University

J. Zico Kolter
Carnegie Mellon University
Bosch Center for AI

Vladlen Koltun
Intel Labs

Bilevel optimization in machine learning

- Mathematical formalism where the problem we want to solve depends on the solution of another problem

- Attention in the ML community because of its ability to model many situations

Model Selection via Bilevel Optimization

Kristin P. Bennett, Jing Hu, Xiaoyun Ji, Gautam Kunapuli, and Jong-Shi Pang

DARTS: DIFFERENTIABLE ARCHITECTURE SEARCH

Hanxiao Liu[*]
CMU
hanxiaol@cs.cmu.com

Karen Simonyan
DeepMind
simonyan@google.com

Yiming Yang
CMU
yiming@cs.cmu.edu

Bilevel Optimization to Learn Training Distributions for Language Modeling under Domain Shift

David Grangier, Pierre Ablin, Awni Hannun
Apple, {grangier,pablin,awni}@apple.com

DADA: Differentiable Automatic Data Augmentation

Yonggang Li[*1], Guosheng Hu[*2,3], Yongtao Wang[†1], Timothy Hospedales[4], Neil M. Robertson[2,3], and Yongxin Yang[4]

Deep Equilibrium Models

Shaojie Bai
Carnegie Mellon University

J. Zico Kolter
Carnegie Mellon University
Bosch Center for AI

Vladlen Koltun
Intel Labs

Meta-Learning with Implicit Gradients

Aravind Rajeswaran[*,1] **Chelsea Finn**[*,2] **Sham Kakade**[1] **Sergey Levine**[2]
[1] University of Washington Seattle [2] University of California Berkeley

Bilevel optimization

Bilevel Optimization Problem

Bilevel optimization

Bilevel Optimization Problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Bilevel optimization

Value function

Bilevel Optimization Problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Bilevel optimization

Value function Outer function

Bilevel Optimization Problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Bilevel optimization

Value function   Outer function

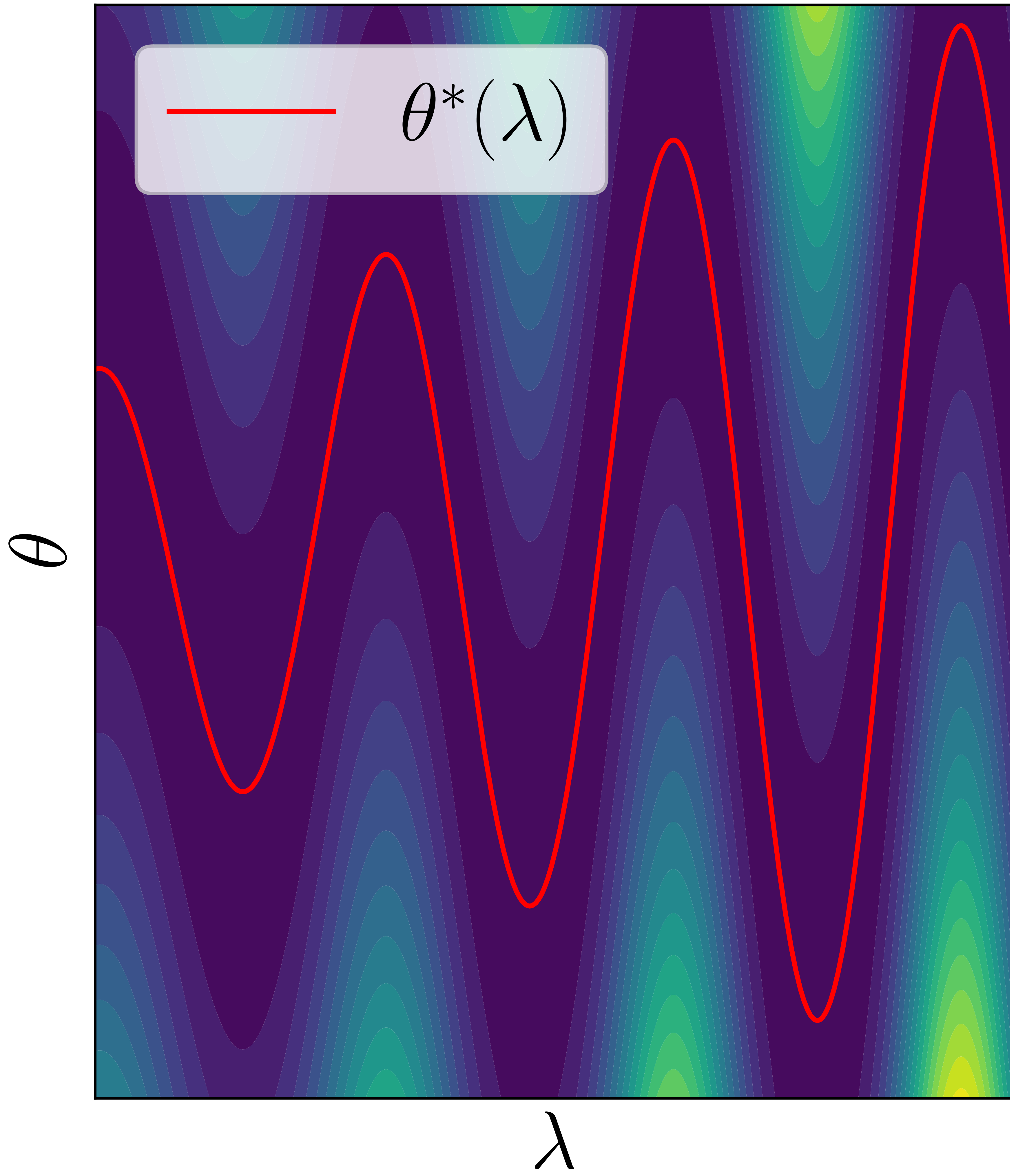
Bilevel Optimization Problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Inner function

Contour of g



Bilevel optimization

Value function

Outer function

Bilevel Optimization Problem

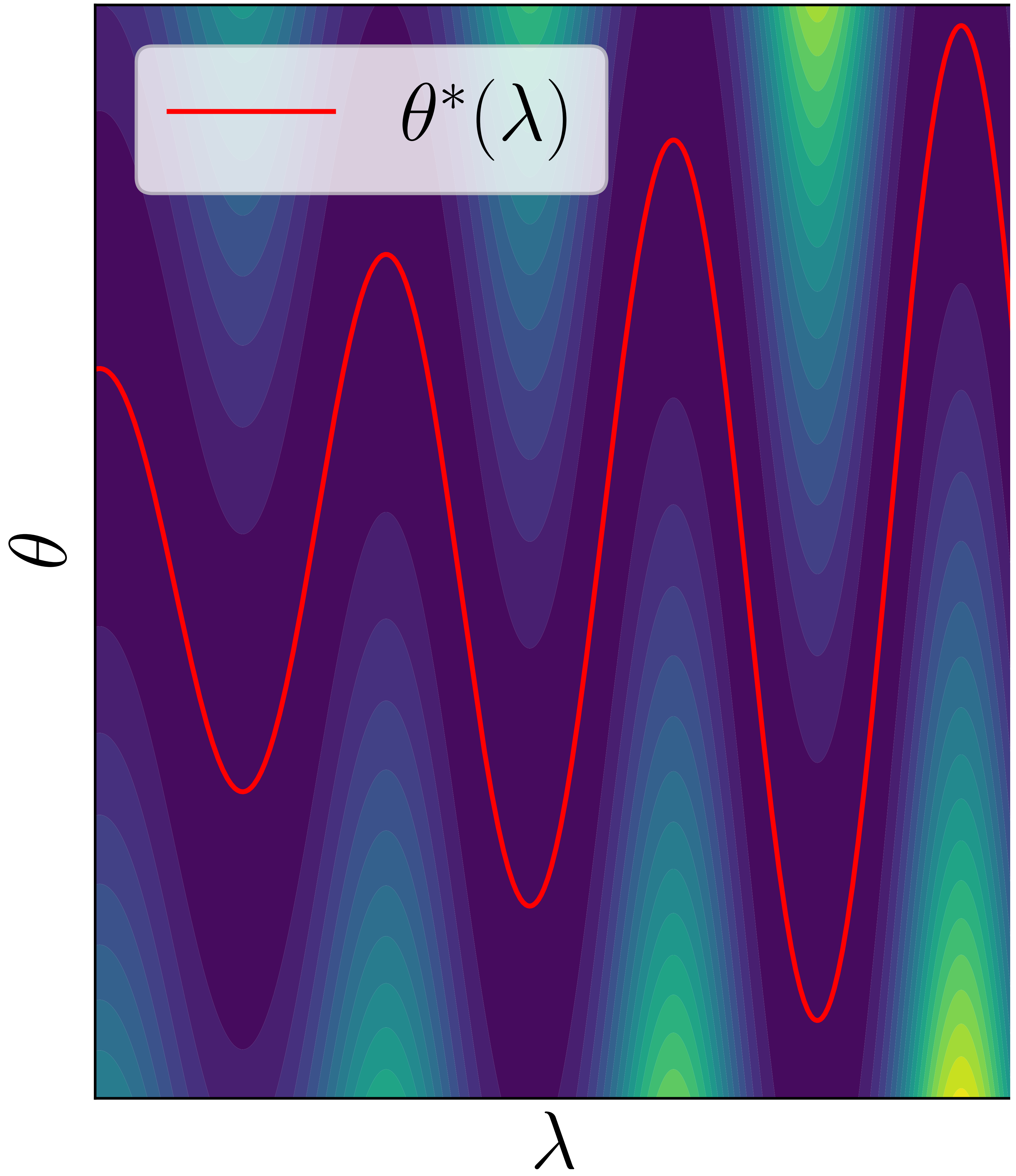
$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Inner variable

Inner function

Contour of g



Bilevel optimization

Value function
Outer function

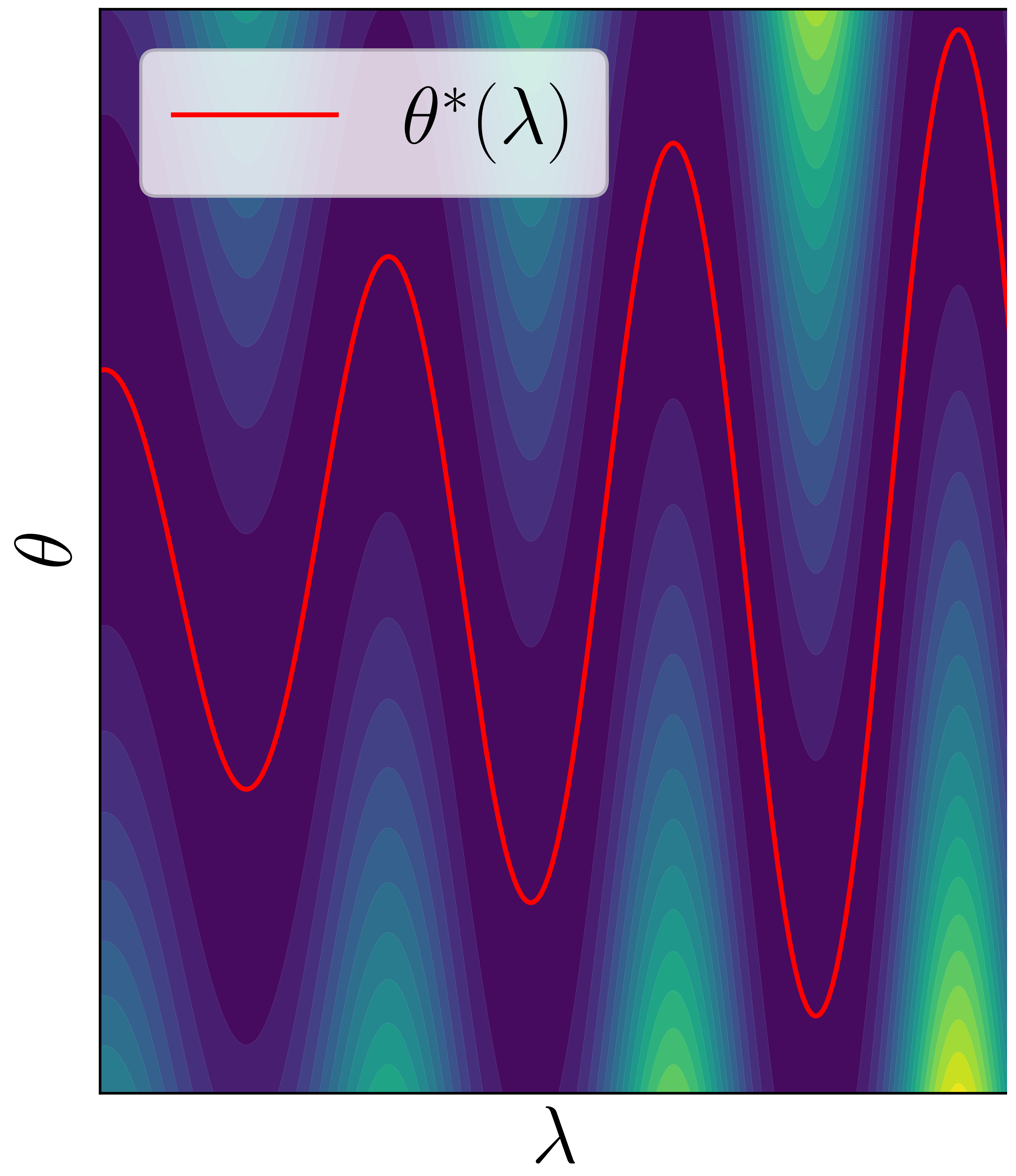
Bilevel Optimization Problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Inner function
Outer variable
Inner variable

Contour of g



Bilevel optimization

Value function

Outer function

Bilevel Optimization Problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

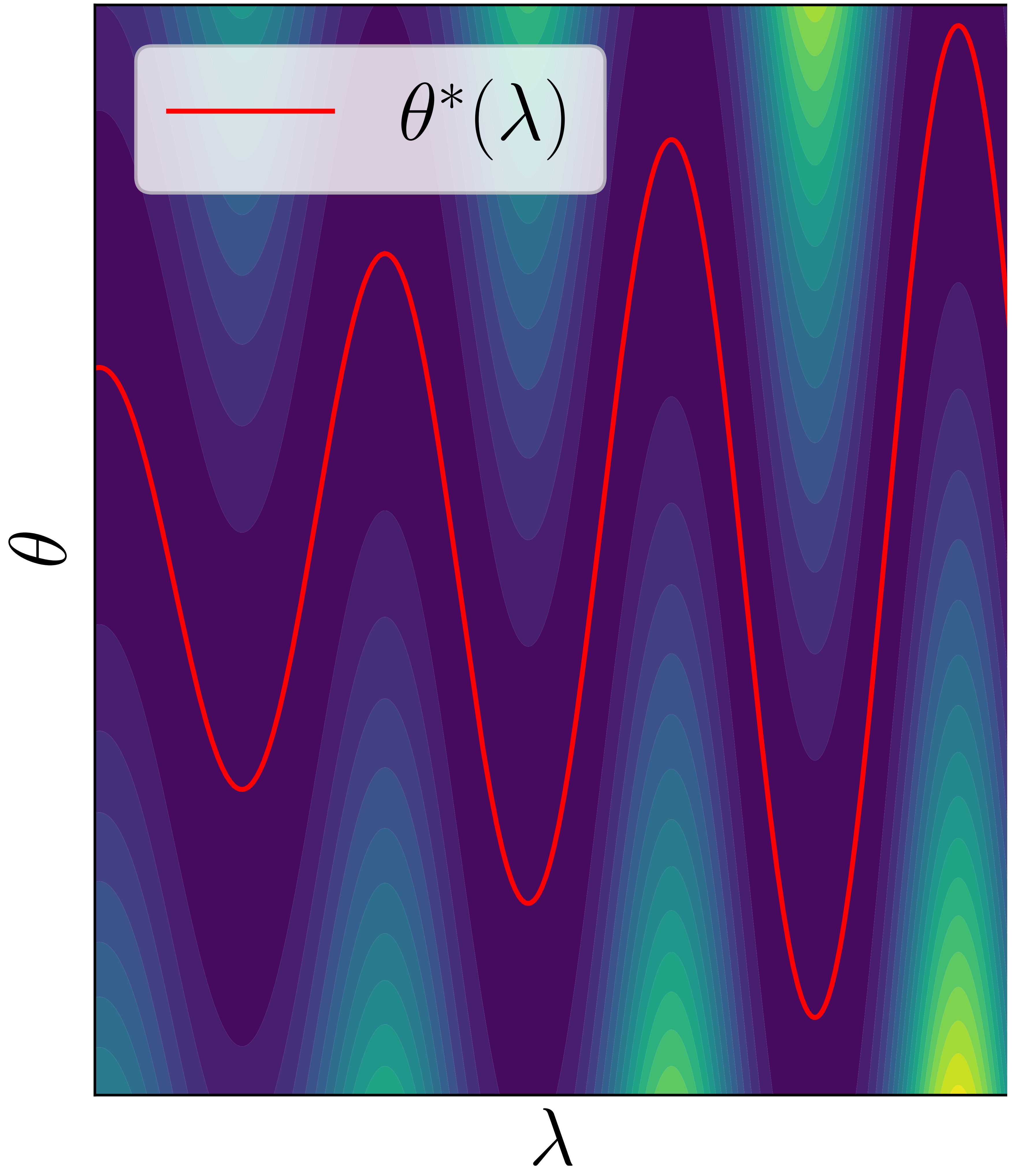
Inner variable

Inner function

Outer variable

- Generally nonconvex eventhough f and g are convex

Contour of g



Bilevel optimization

Value function Outer function

Bilevel Optimization Problem

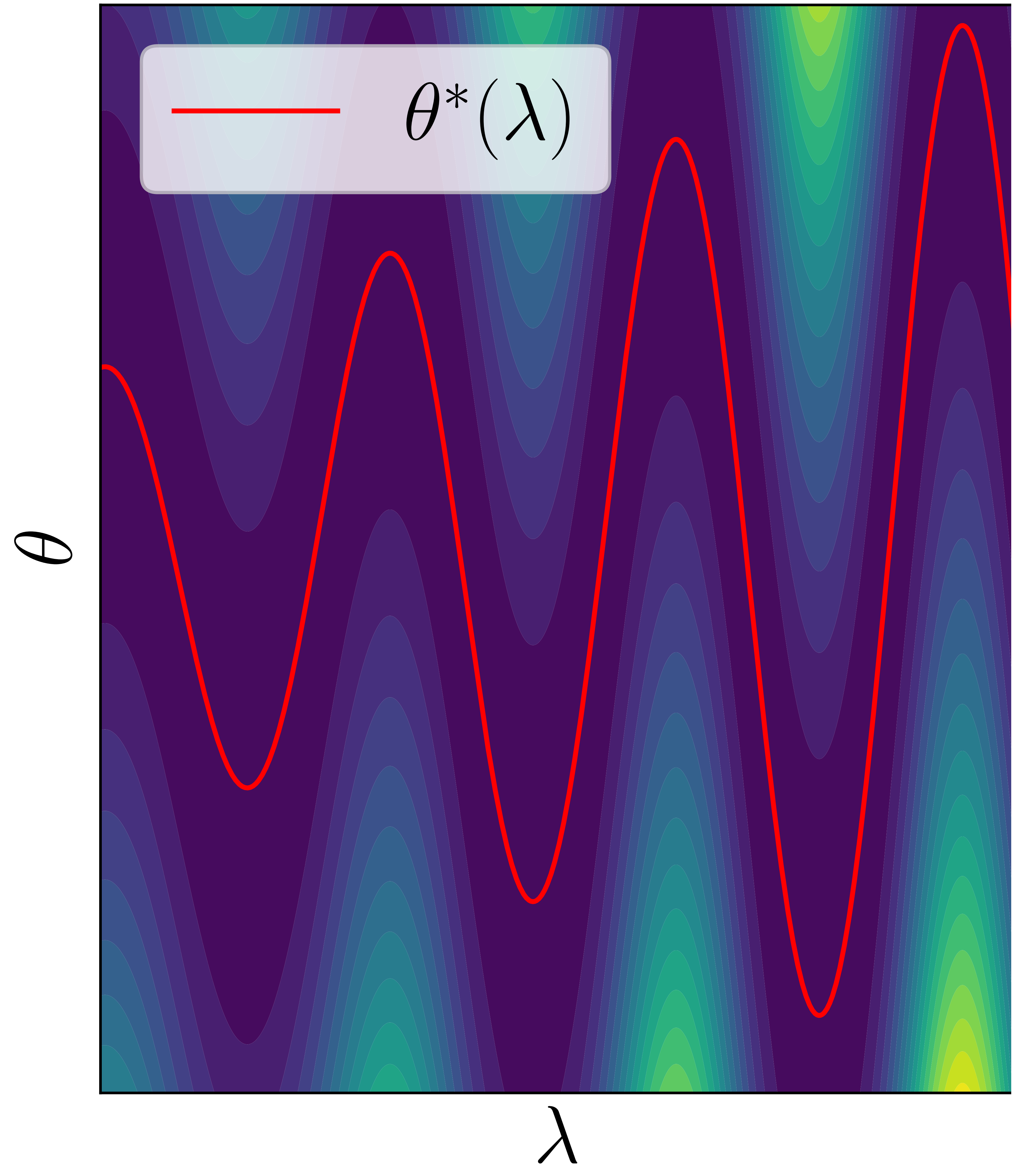
$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Inner function Outer variable Inner variable

- Generally nonconvex eventhough f and g are convex
- This definition assumes the uniqueness of the inner solution

Contour of g



Bilevel optimization

Value function

Outer function

Bilevel Optimization Problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

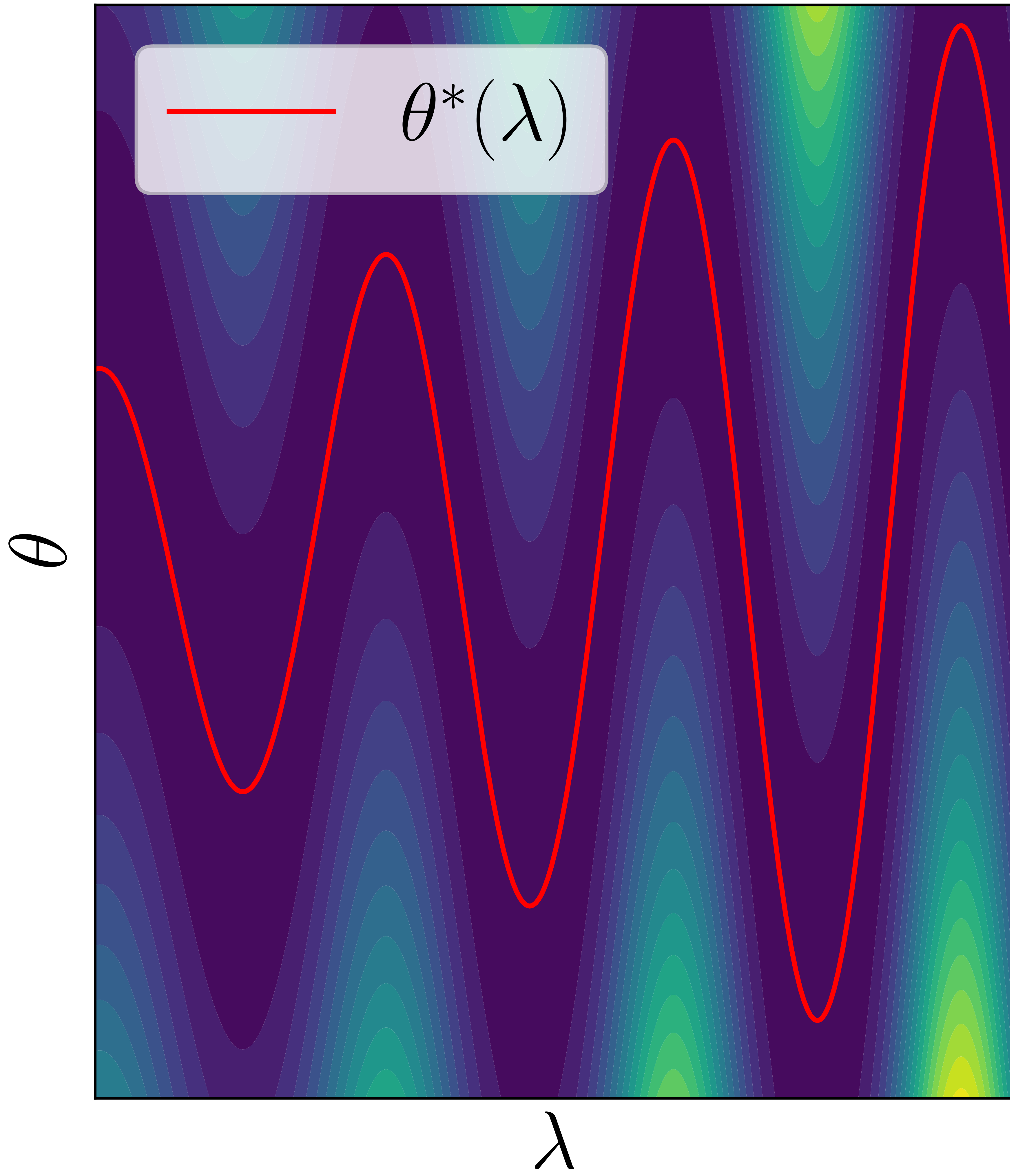
Inner variable

Inner function

Outer variable

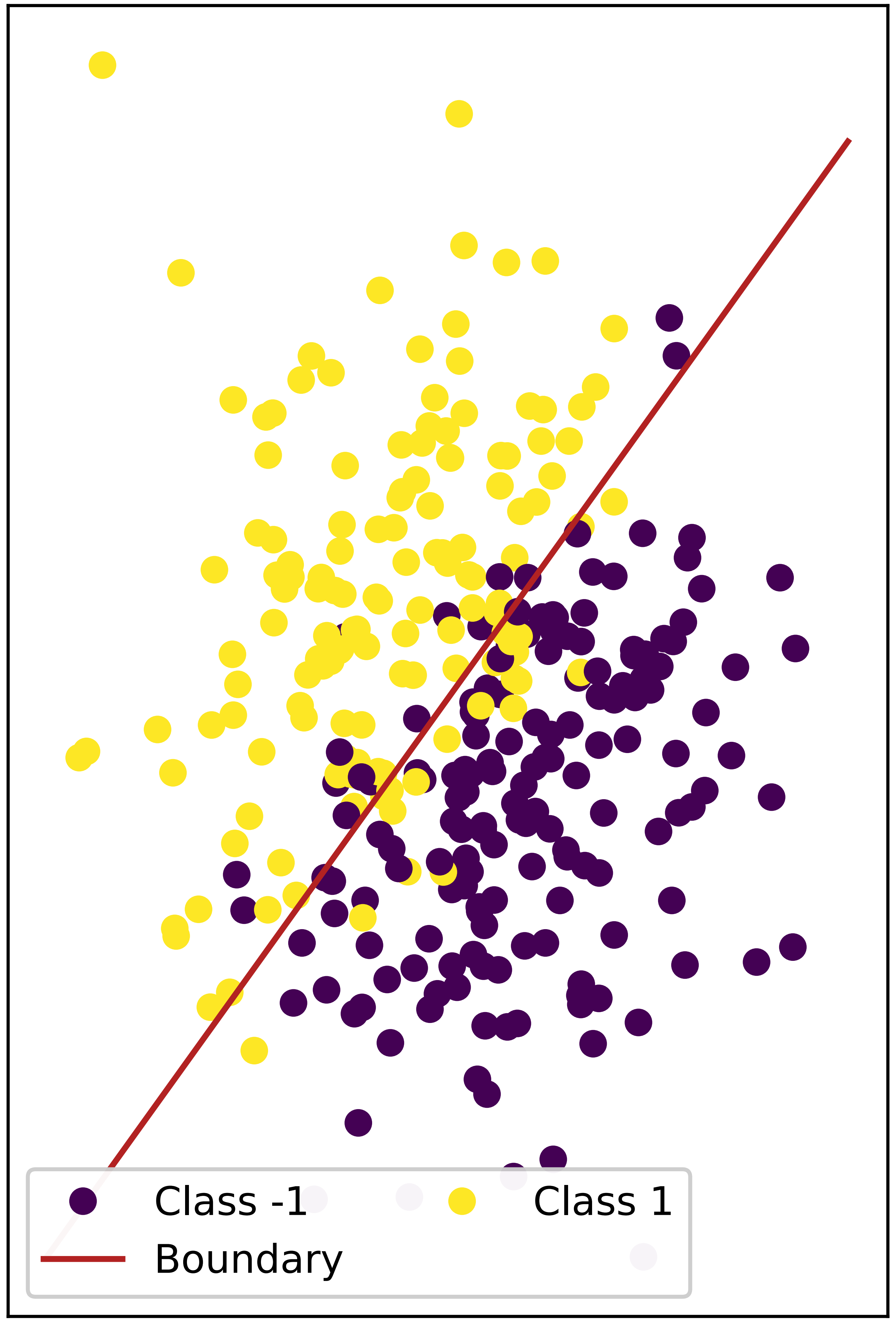
- Generally nonconvex eventhough f and g are convex
- This definition assumes the uniqueness of the inner solution
- Non-uniqueness leads to dramatically hard problems[Bolte et al. ’24]

Contour of g



Hyperparameter selection[Larsen ’96, Bennett et al. ’06]

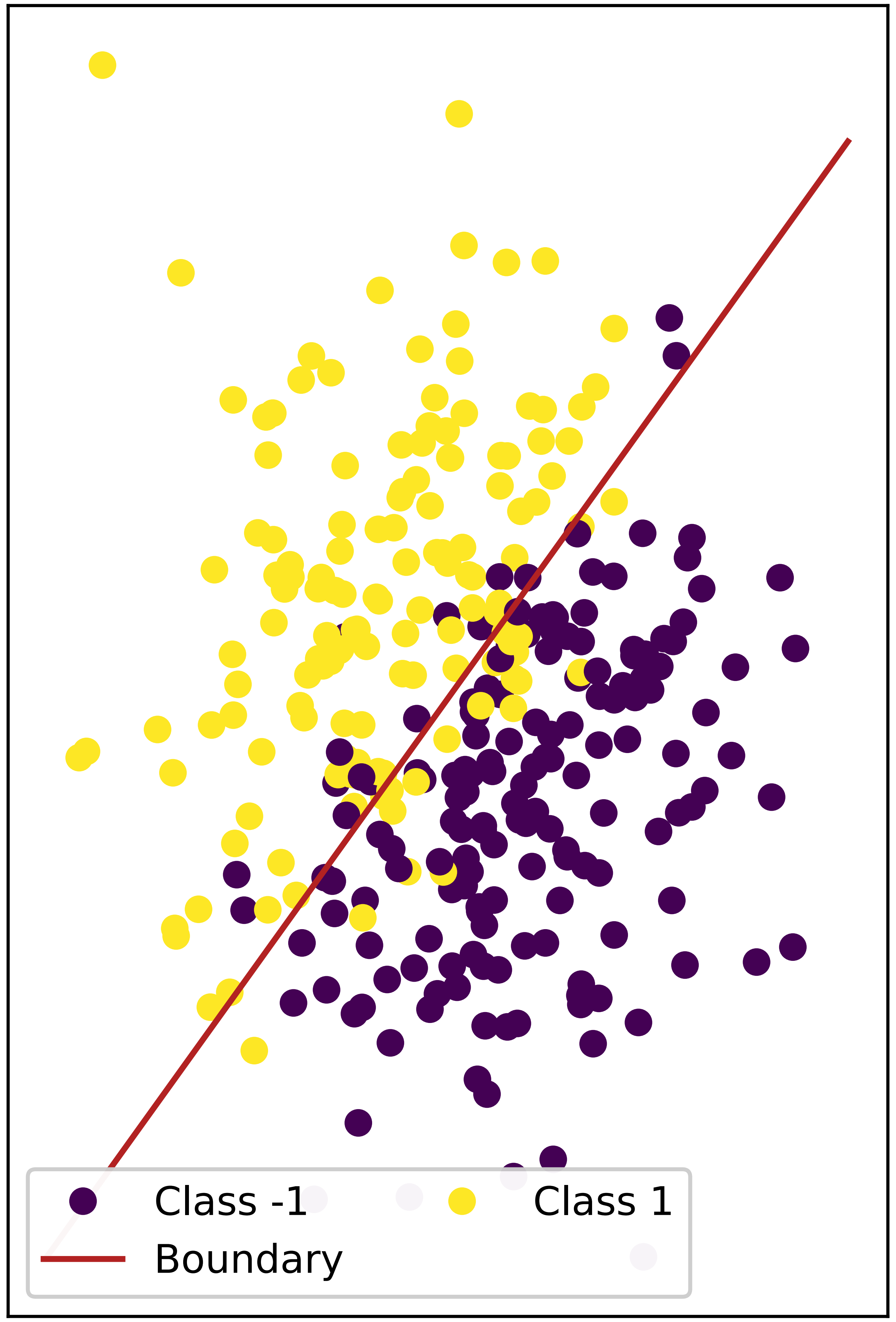
Learning = Solving an optimization problem



Hyperparameter selection [Larsen ’96, Bennett et al. ’06]

Learning = Solving an optimization problem

- Training samples $\{(x_i^{\text{train}}, y_i^{\text{train}})\}_{i=1}^n$, prediction function $(h_\theta)_{\theta \in \mathbb{R}^{d_\theta}}$.

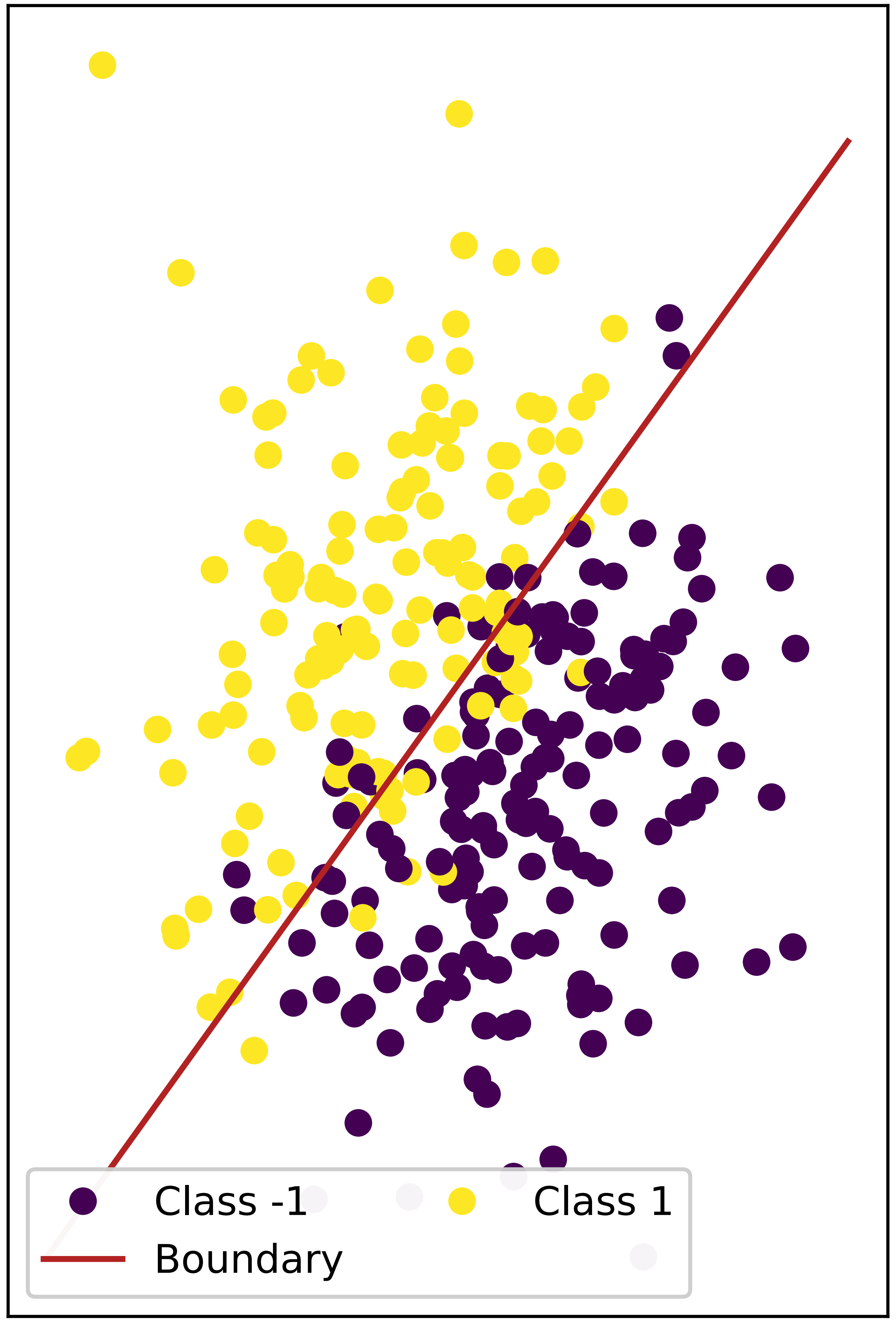


Hyperparameter selection [Larsen ’96, Bennett et al. ’06]

Learning = Solving an optimization problem

- Training samples $\{(x_i^{\text{train}}, y_i^{\text{train}})\}_{i=1}^n$, prediction function $(h_\theta)_{\theta \in \mathbb{R}^{d_\theta}}$.
- Empirical Risk Minimization

$$\min_{\theta \in \mathbb{R}^{d_\theta}} g(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i^{\text{train}}, h_\theta(x_i^{\text{train}}))$$



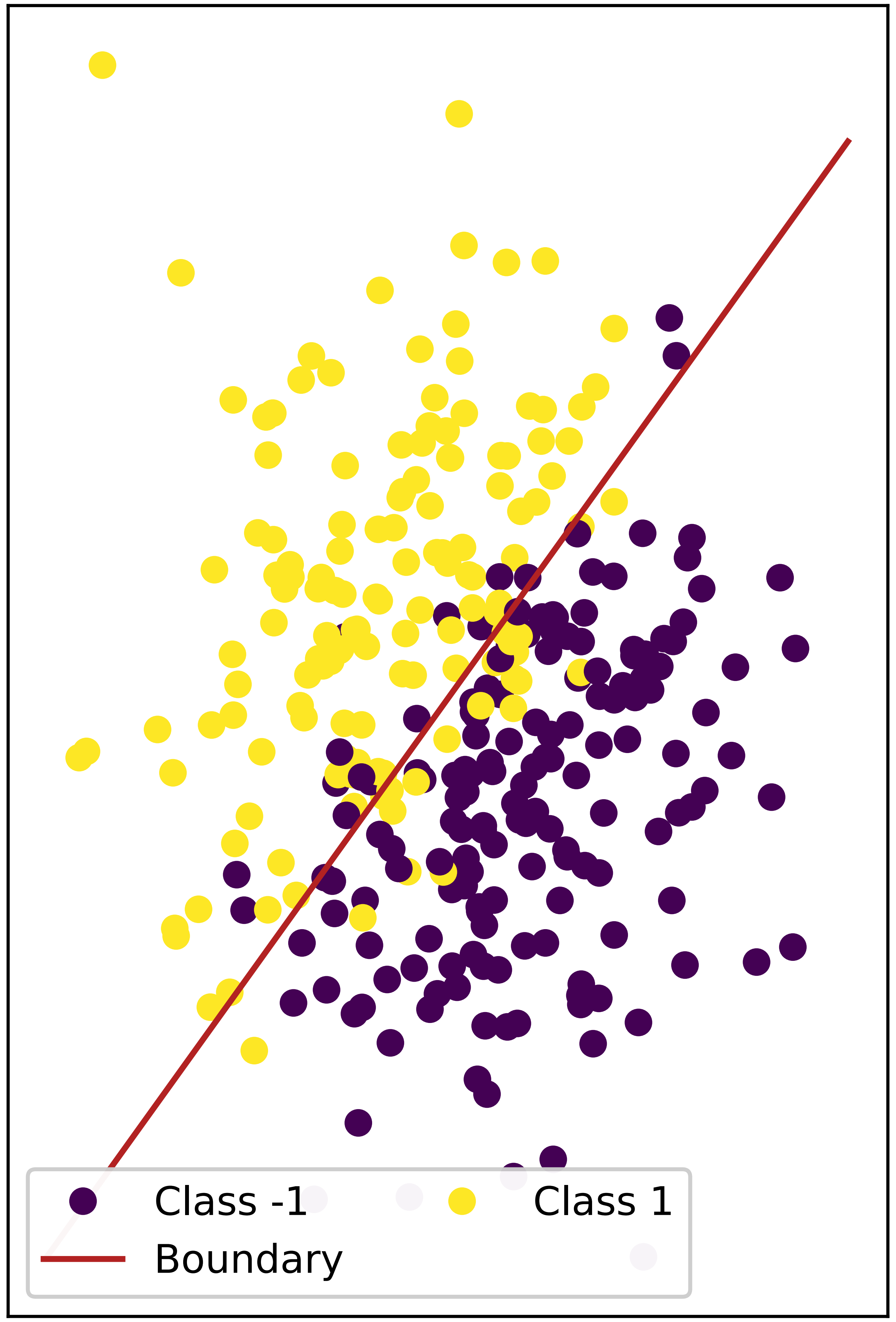
Hyperparameter selection[Larsen '96, Bennett et al. '06]

Learning = Solving an optimization problem

- Training samples $\{(x_i^{\text{train}}, y_i^{\text{train}})\}_{i=1}^n$, prediction function $(h_\theta)_{\theta \in \mathbb{R}^{d_\theta}}$.
- Empirical Risk Minimization

$$\min_{\theta \in \mathbb{R}^{d_\theta}} g(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i^{\text{train}}, h_\theta(x_i^{\text{train}}))$$

- Solution θ^* found by running SGD[Robbins & Monro '54]



Hyperparameter selection[Larsen '96, Bennett et al. '06]

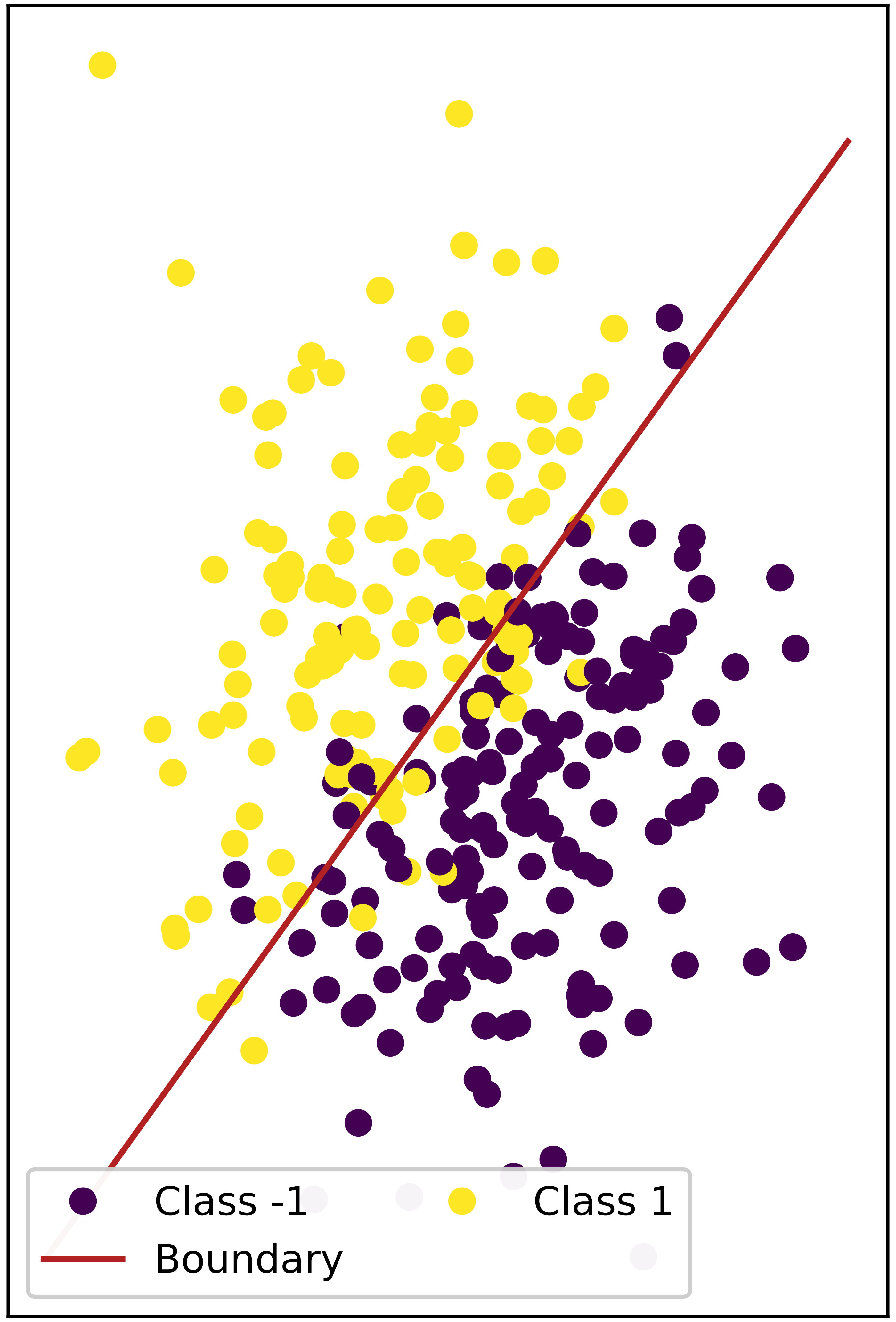
Learning = Solving an optimization problem

- Training samples $\{(x_i^{\text{train}}, y_i^{\text{train}})\}_{i=1}^n$, prediction function $(h_\theta)_{\theta \in \mathbb{R}^{d_\theta}}$.
- Empirical Risk Minimization

$$\min_{\theta \in \mathbb{R}^{d_\theta}} g(\theta) = \frac{1}{n} \sum_{i=1}^n \ell(y_i^{\text{train}}, h_\theta(x_i^{\text{train}}))$$

- Solution θ^* found by running SGD[Robbins & Monro '54]
- There are hyperparameters, e.g. regularization

$$\min_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^n \ell(y_i^{\text{train}}, h_\theta(x_i^{\text{train}})) + \frac{\lambda}{2} \|\theta\|^2$$



Hyperparameter selection[Larsen ’96, Bennett et al. ’06]

Learning = Solving an optimization problem

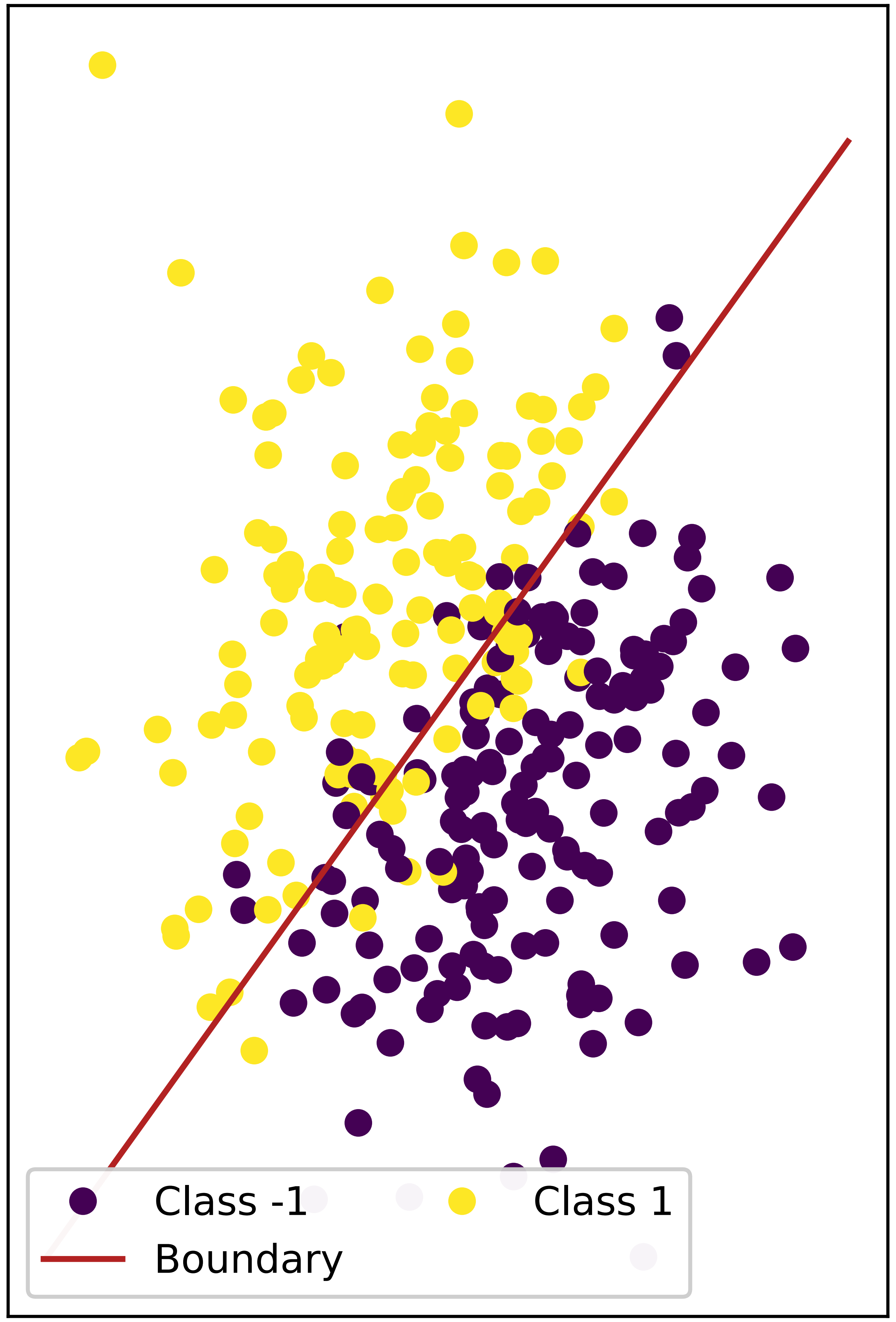
- Training samples $\{(x_i^{\text{train}}, y_i^{\text{train}})\}_{i=1}^n$, prediction function $(h_\theta)_{\theta \in \mathbb{R}^{d_\theta}}$.
- Empirical Risk Minimization

$$\min_{\theta \in \mathbb{R}^{d_\theta}} g(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i^{\text{train}}, h_\theta(x_i^{\text{train}}))$$

- Solution θ^* found by running SGD[Robbins & Monro ’54]
- There are hyperparameters, e.g. regularization

$$\min_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i^{\text{train}}, h_\theta(x_i^{\text{train}})) + \frac{\lambda}{2}\|\theta\|^2$$

- The learnt parameter $\theta^*(\lambda)$ depends on λ



Hyperparameter selection [Larsen '96, Bennett et al. '06]

Learning = Solving an optimization problem

- Training samples $\{(x_i^{\text{train}}, y_i^{\text{train}})\}_{i=1}^n$, prediction function $(h_\theta)_{\theta \in \mathbb{R}^{d_\theta}}$.
- Empirical Risk Minimization

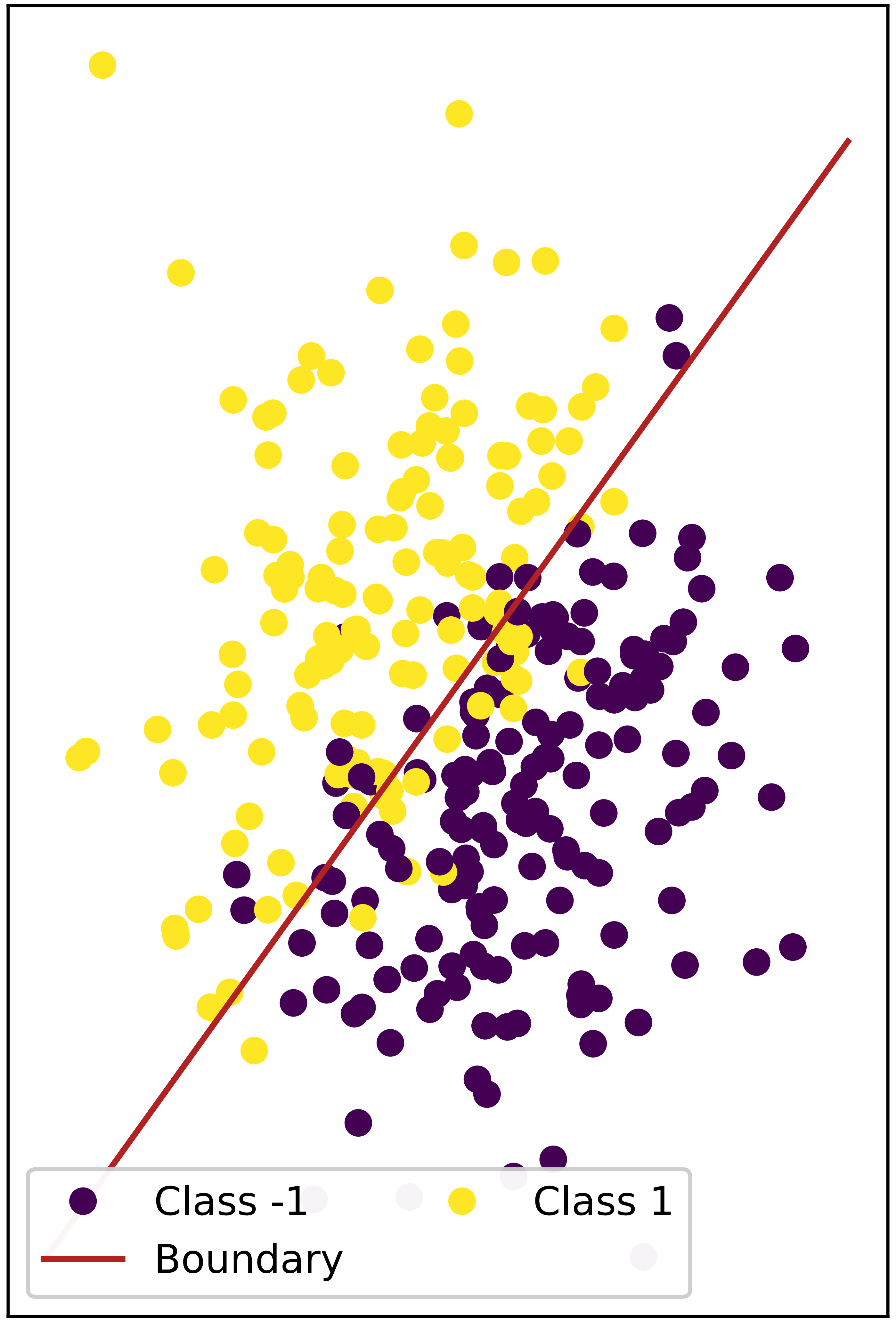
$$\min_{\theta \in \mathbb{R}^{d_\theta}} g(\theta) = \frac{1}{n} \sum_{i=1}^n \ell(y_i^{\text{train}}, h_\theta(x_i^{\text{train}}))$$

- Solution θ^* found by running SGD [Robbins & Monro '54]
- There are hyperparameters, e.g. regularization

$$\min_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^n \ell(y_i^{\text{train}}, h_\theta(x_i^{\text{train}})) + \frac{\lambda}{2} \|\theta\|^2$$

- The learnt parameter $\theta^*(\lambda)$ depends on λ
- λ selected by minimizing the validation loss

$$f(\theta^*(\lambda)) = \frac{1}{m} \sum_{j=1}^m \ell(y_i^{\text{val}}, h_{\theta^*(\lambda)}(x_i^{\text{val}}))$$



Hyperparameter selection [Larsen '96, Bennett et al. '06]

Learning = Solving an optimization problem

- Training samples $\{(x_i^{\text{train}}, y_i^{\text{train}})\}_{i=1}^n$, prediction function $(h_\theta)_{\theta \in \mathbb{R}^{d_\theta}}$.
- Empirical Risk Minimization

$$\min_{\theta \in \mathbb{R}^{d_\theta}} g(\theta) = \frac{1}{n} \sum_{i=1}^n \ell(y_i^{\text{train}}, h_\theta(x_i^{\text{train}}))$$

- Solution θ^* found by running SGD [Robbins & Monro '54]
- There are hyperparameters, e.g. regularization

$$\min_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^n \ell(y_i^{\text{train}}, h_\theta(x_i^{\text{train}})) + \frac{\lambda}{2}\|\theta\|^2$$

- The learnt parameter $\theta^*(\lambda)$ depends on λ
- λ selected by minimizing the validation loss

$$f(\theta^*(\lambda)) = \frac{1}{m} \sum_{j=1}^m \ell(y_i^{\text{val}}, h_{\theta^*(\lambda)}(x_i^{\text{val}}))$$

Bilevel problem

$$\min_\lambda f(\theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Solving bilevel problems

Zeroth-order methods

Grid search

Zeroth-order methods

Grid search

1. Define a grid of candidates $\lambda_1, \ldots, \lambda_K$

Zeroth-order methods

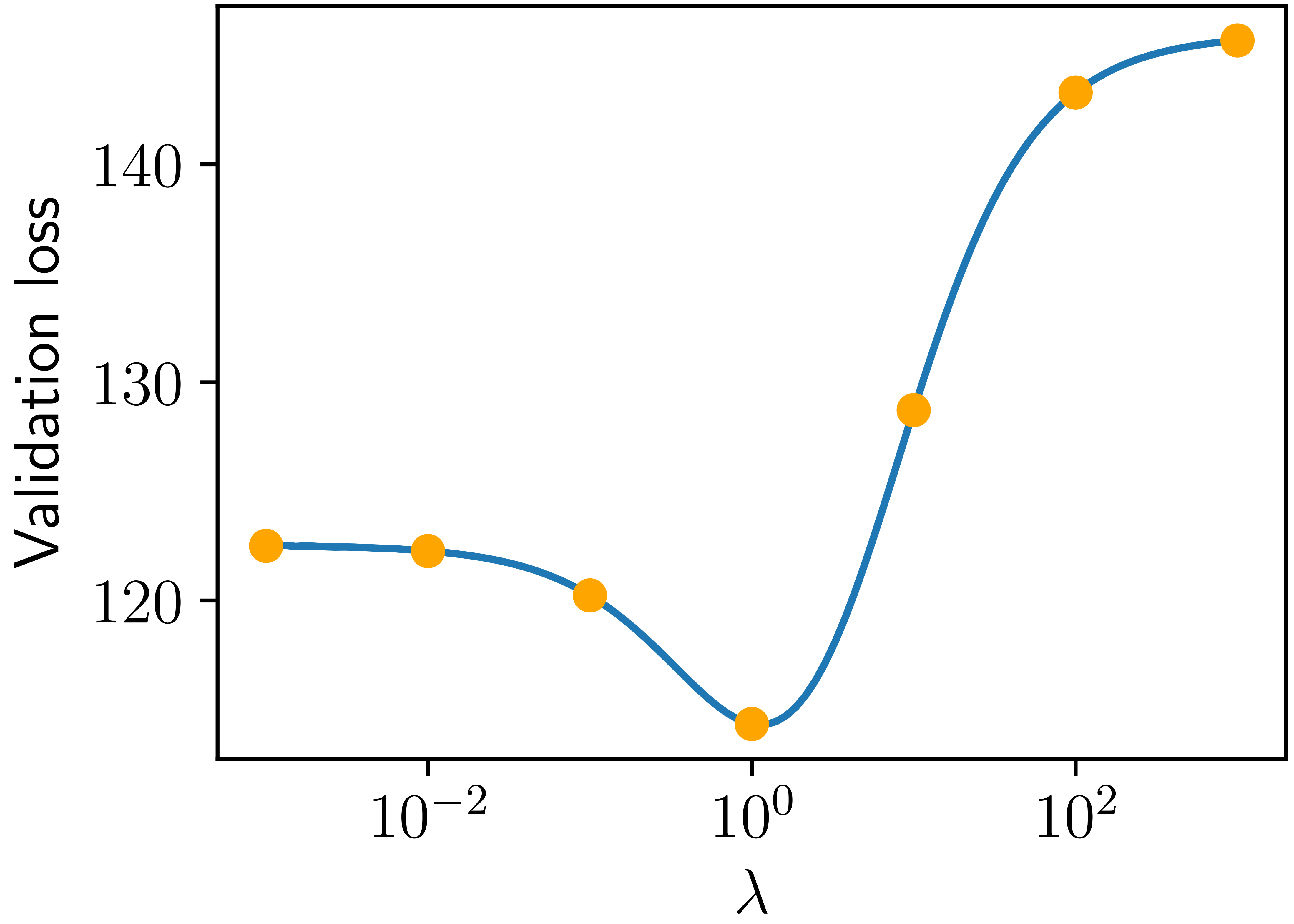
Grid search

1. Define a grid of candidates $\lambda_1, \ldots, \lambda_K$
2. Train the model to get $\theta^*(\lambda_1), \ldots, \theta^*(\lambda_K)$

Zeroth-order methods

Grid search

1. Define a grid of candidates $\lambda_1, \ldots, \lambda_K$
2. Train the model to get $\theta^*(\lambda_1), \ldots, \theta^*(\lambda_K)$
3. Select the one that minimizes the value function $\Phi(\lambda) = f(\lambda, \theta^*(\lambda))$

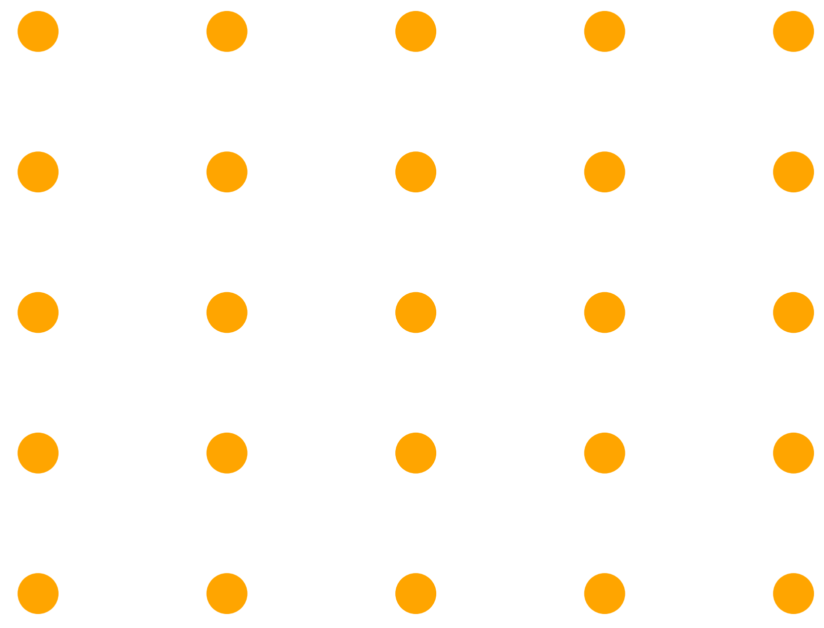


The problem with the grid search

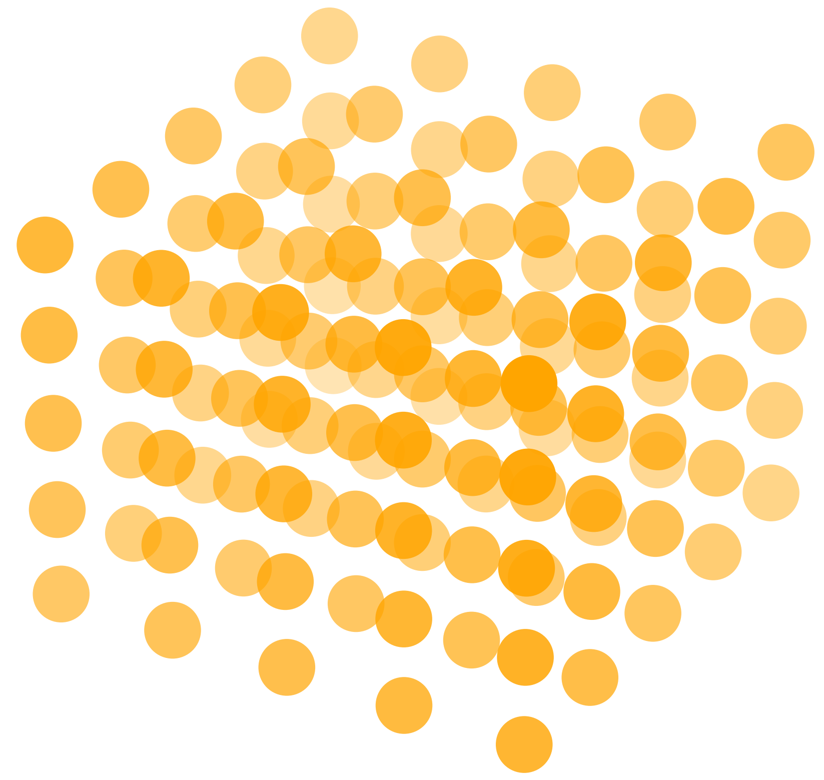
Grid search

1. Define a grid of candidates $\lambda_1, \ldots, \lambda_K$
2. Train the model to get $\theta^*(\lambda_1), \ldots, \theta^*(\lambda_K)$
3. Select the one that minimizes the value function

$$\left|\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}\right| = 5$$



$$\left|\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}^2\right| = 5^2 = 25$$



$$\left|\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}^3\right| = 5^3 = 125$$

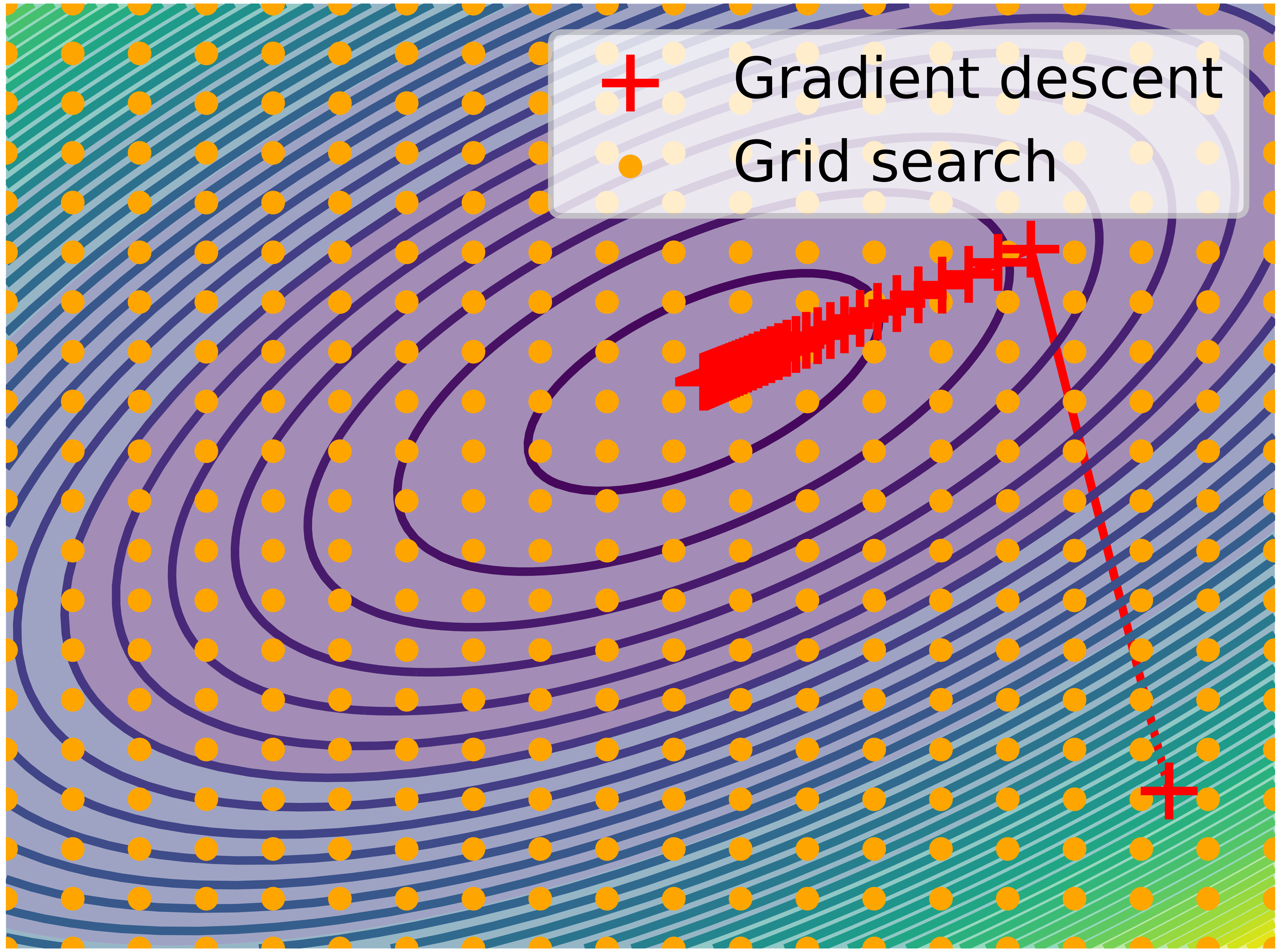
Curse of dimensionality

The number of function evaluations scales exponentially with the dimension

First-order optimization

Gradient descent on Φ

$$\lambda^{t+1} = \lambda^t - \gamma \nabla \Phi(\lambda^t)$$

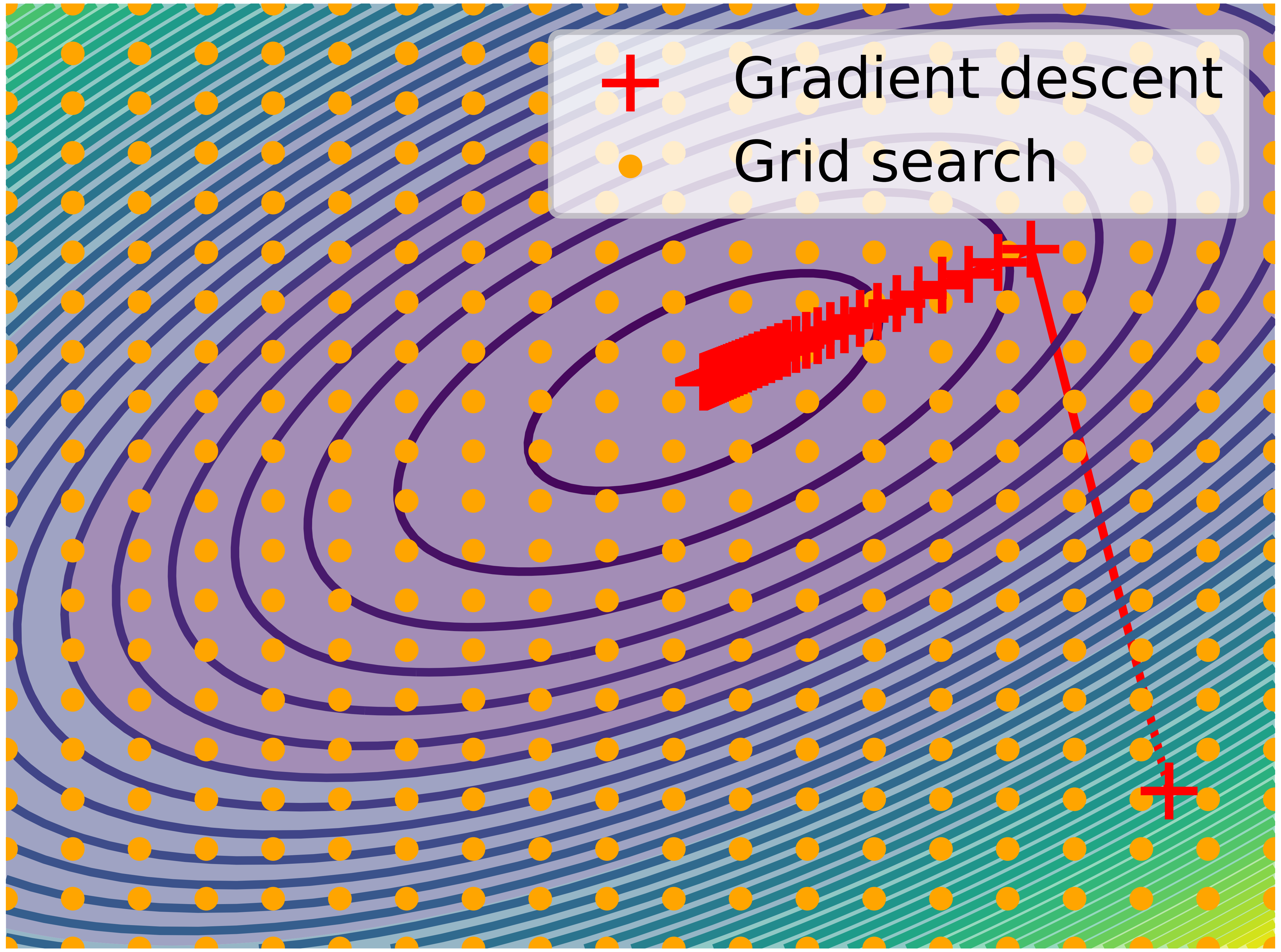


First-order optimization

Gradient descent on Φ

$$\lambda^{t+1} = \lambda^t - \gamma \nabla \Phi(\lambda^t)$$

Complexity



First-order optimization

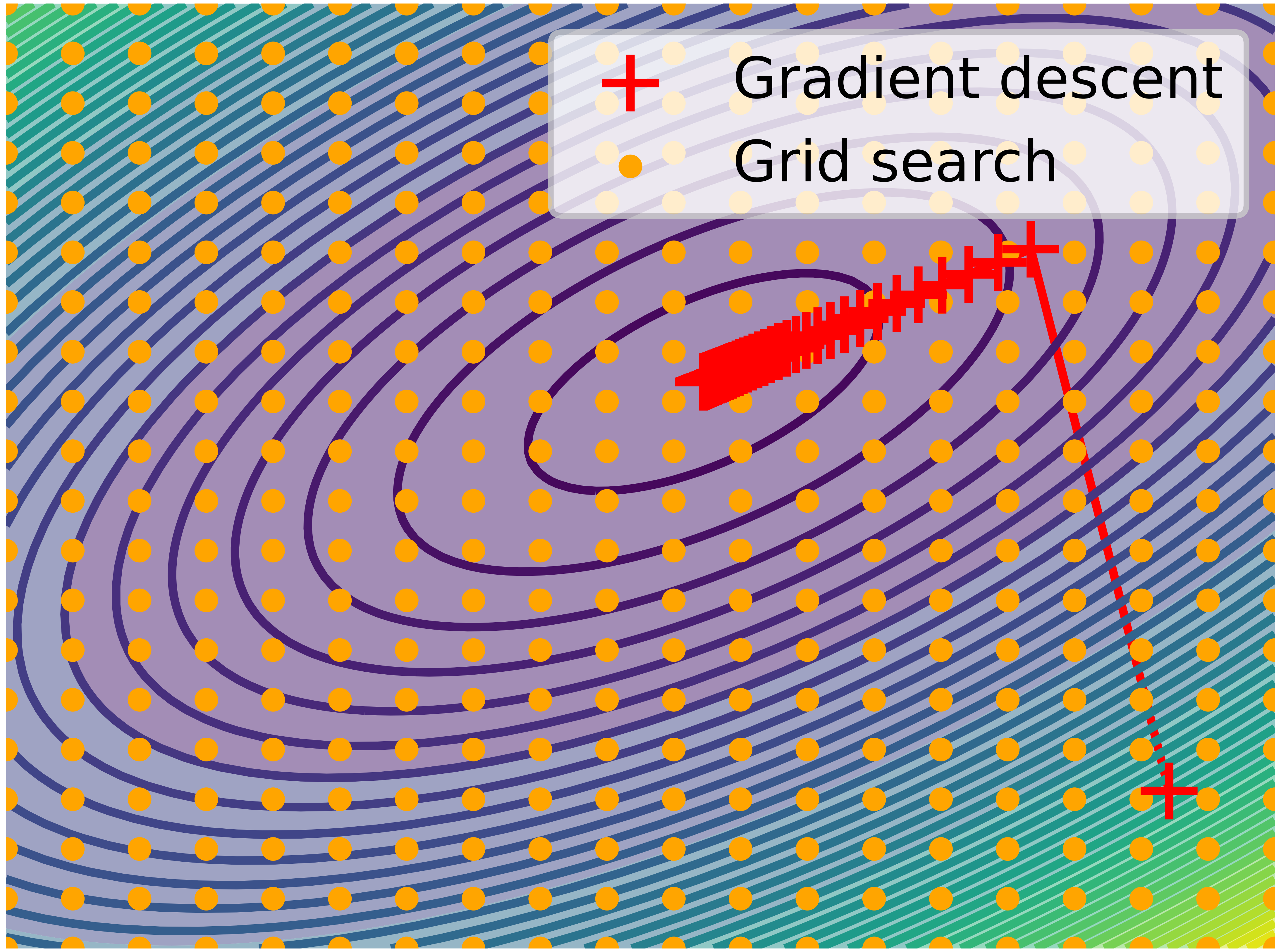
Gradient descent on Φ

$$\lambda^{t+1} = \lambda^t - \gamma \nabla \Phi(\lambda^t)$$

Complexity

✓ Number of gradient computations to reach an ϵ-stationary point if Φ is smooth:

$$\mathcal{O}(\epsilon^{-1})$$



First-order optimization

Gradient descent on Φ

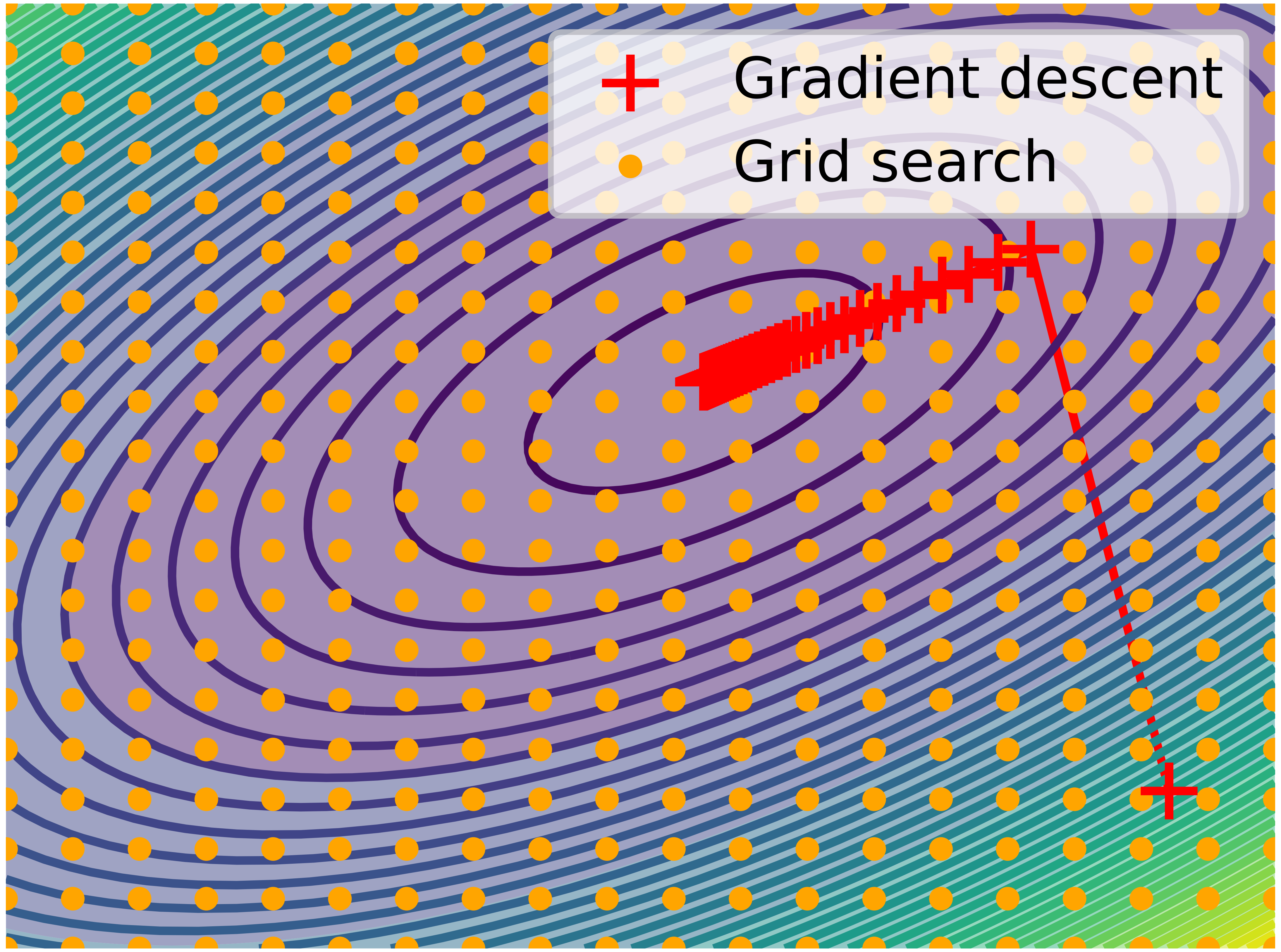
$$\lambda^{t+1} = \lambda^t - \gamma \nabla \Phi(\lambda^t)$$

Complexity

✓ Number of gradient computations to reach an ϵ-stationary point if Φ is smooth:

$$\mathcal{O}(\epsilon^{-1})$$

Independent from the input dimension



First-order optimization

Differentiable?

Gradient descent on Φ

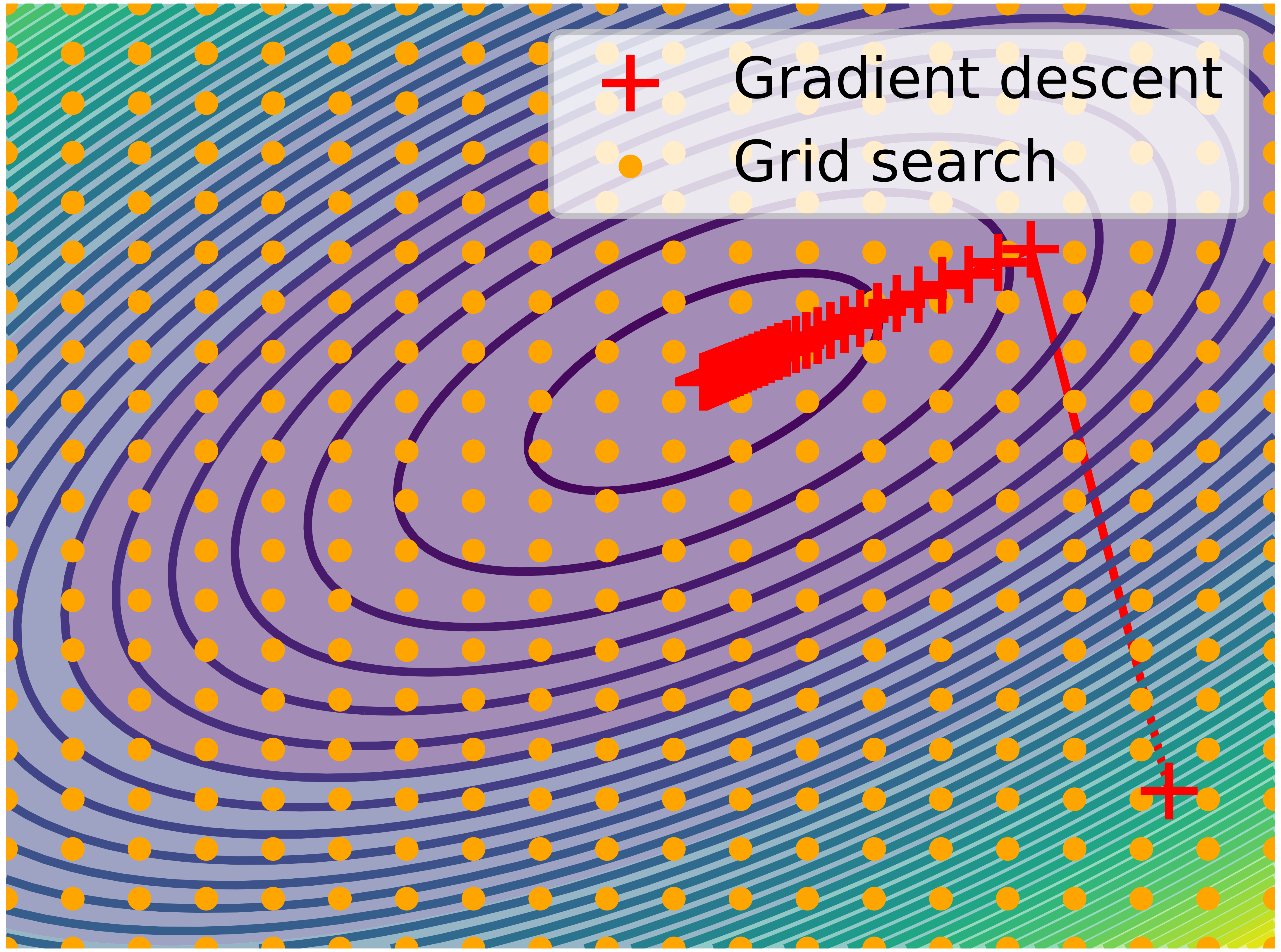
$$\lambda^{t+1} = \lambda^t - \gamma \nabla \Phi(\lambda^t)$$

Complexity

✓ Number of gradient computations to reach an ϵ-stationary point if Φ is smooth:

$$\mathcal{O}(\epsilon^{-1})$$

Independent from the input dimension



First-order optimization

Differentiable?

Gradient descent on Φ

$$\lambda^{t+1} = \lambda^t - \gamma \nabla \Phi(\lambda^t)$$

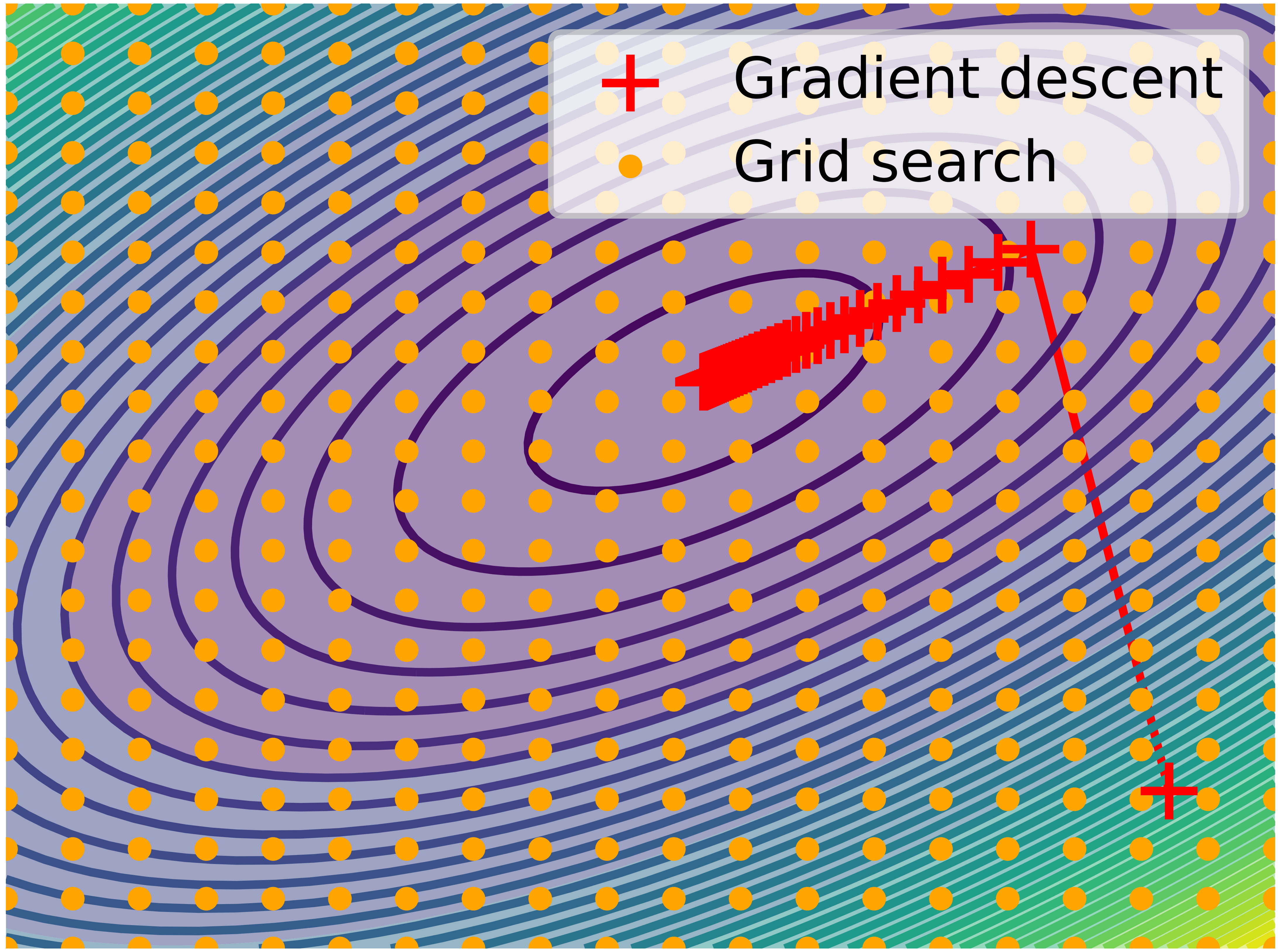
How to compute it?

Complexity

✓ Number of gradient computations to reach an ϵ-stationary point if Φ is smooth:

$$\mathcal{O}(\epsilon^{-1})$$

Independent from the input dimension



Implicit differentiation [Jongen et al. ’90, Dempe ‘93, Larsen ’96, Dempe ’98]

Bilevel problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Implicit differentiation [Jongen et al. ’90, Dempe ‘93, Larsen ’96, Dempe ’98]

Differentiable

Bilevel problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Implicit differentiation [Jongen et al. ’90, Dempe ‘93, Larsen ’96, Dempe ’98]

Differentiable

Bilevel problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Twice differentiable and strongly convex

Implicit differentiation [Jongen et al. ’90, Dempe ‘93, Larsen ’96, Dempe ’98]

Differentiable

Bilevel problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Twice differentiable and strongly convex

Chain rule

$$\nabla \Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) + \mathrm{d}\theta^*(\lambda)^\top \nabla_\theta f(\lambda, \theta^*(\lambda))$$

Implicit differentiation [Jongen et al. '90, Dempe '93, Larsen '96, Dempe '98]

Differentiable

Bilevel problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \underset{\theta \in \mathbb{R}^{d_\theta}}{\operatorname{argmin}}\, g(\lambda, \theta)$$

Twice differentiable and strongly convex

First-order condition

$$\nabla_\theta g(\lambda, \theta^*(\lambda)) = 0$$

Chain rule

$$\nabla \Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) + \mathrm{d}\theta^*(\lambda)^\top \nabla_\theta f(\lambda, \theta^*(\lambda))$$

Implicit differentiation [Jongen et al. '90, Dempe '93, Larsen '96, Dempe '98]

Differentiable

Bilevel problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Twice differentiable and strongly convex

First-order condition

$$\nabla_\theta g(\lambda, \theta^*(\lambda)) = 0$$

$$\frac{\partial}{\partial \lambda}$$

$$\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda)) \mathrm{d}\theta^*(\lambda) + \nabla^2_{\theta,\lambda} g(\lambda, \theta^*(\lambda)) = 0$$

Chain rule

$$\nabla \Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) + \mathrm{d}\theta^*(\lambda)^\top \nabla_\theta f(\lambda, \theta^*(\lambda))$$

Implicit differentiation [Jongen et al. '90, Dempe '93, Larsen '96, Dempe '98]

Differentiable

Bilevel problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Twice differentiable and strongly convex

First-order condition

$$\nabla_\theta g(\lambda, \theta^*(\lambda)) = 0$$

$$\Big\downarrow \frac{\partial}{\partial \lambda}$$

$$\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda)) \mathrm{d}\theta^*(\lambda) + \nabla^2_{\theta,\lambda} g(\lambda, \theta^*(\lambda)) = 0$$

$$\mathrm{d}\theta^*(\lambda) = -\left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla^2_{\theta,\lambda} g(\lambda, \theta^*(\lambda))$$

Chain rule

$$\nabla \Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) + \mathrm{d}\theta^*(\lambda)^\top \nabla_\theta f(\lambda, \theta^*(\lambda))$$

Implicit differentiation [Jongen et al. ’90, Dempe ‘93, Larsen ’96, Dempe ’98]

Differentiable

Bilevel problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) = \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

strongly convex

First-order condition

$$\nabla_\theta g(\lambda, \theta^*(\lambda)) = 0$$

$$\frac{\partial}{\partial \lambda}$$

$$\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda)) \mathrm{d}\theta^*(\lambda) + \nabla^2_{\theta,\lambda} g(\lambda, \theta^*(\lambda)) = 0$$

$$\mathrm{d}\theta^*(\lambda) = -\left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla^2_{\theta,\lambda} g(\lambda, \theta^*(\lambda))$$

Chain rule

$$\nabla \Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) + \mathrm{d}\theta^*(\lambda)^\top \nabla_\theta f(\lambda, \theta^*(\lambda))$$

$$\nabla \Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) - \nabla^2_{\lambda,\theta} g(\lambda, \theta^*(\lambda)) \left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla_\theta f(\lambda, \theta^*(\lambda))$$

Implicit differentiation in practice

Implicit gradient

$$\nabla\Phi(\lambda) = \nabla_\lambda f(\lambda,\ \theta^*(\lambda)\ ) - \nabla^2_{\lambda,\theta} g(\lambda,\ \theta^*(\lambda)\ )\left[\nabla^2_{\theta,\theta} g(\lambda,\ \theta^*(\lambda)\ )\right]^{-1}\nabla_\theta f(\lambda,\ \theta^*(\lambda)\ )$$

Implicit differentiation in practice

Implicit gradient

$$\nabla\Phi(\lambda) = \nabla_\lambda f(\lambda,\ \theta^*(\lambda)\ ) - \nabla^2_{\lambda,\theta} g(\lambda,\ \theta^*(\lambda)\ ) \left[\nabla^2_{\theta,\theta} g(\lambda,\ \theta^*(\lambda)\ )\right]^{-1} \nabla_\theta f(\lambda,\ \theta^*(\lambda)\ )$$

Bottlenecks

Implicit differentiation in practice

Implicit gradient

$$\nabla \Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) - \nabla^2_{\lambda,\theta} g(\lambda, \theta^*(\lambda)) \left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla_\theta f(\lambda, \theta^*(\lambda))$$

Bottlenecks

- Solution of the inner problem

Implicit differentiation in practice

Implicit gradient

$$\nabla\Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) - \nabla^2_{\lambda,\theta} g(\lambda, \theta^*(\lambda)) \boxed{\left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla_\theta f(\lambda, \theta^*(\lambda))}$$

Bottlenecks

- Solution of the inner problem
- Solution of a linear system

Implicit differentiation in practice

Implicit gradient

$$\nabla\Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) - \nabla^2_{\lambda,\theta} g(\lambda, \theta^*(\lambda)) \boxed{\left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla_\theta f(\lambda, \theta^*(\lambda))}$$

Bottlenecks

- Solution of the inner problem
- Solution of a linear system
- Computing a gradient is expensive

Implicit differentiation in practice

Implicit gradient

$$\nabla\Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) - \nabla^2_{\lambda,\theta} g(\lambda, \theta^*(\lambda)) \boxed{\left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla_\theta f(\lambda, \theta^*(\lambda))}$$

Bottlenecks

- Solution of the inner problem
- Solution of a linear system
- Computing a gradient is expensive

ML setting: Empirical Risk Minimization

$$f(\lambda, \theta) = \frac{1}{m}\sum_{j=1}^{m} f_j(\lambda, \theta), \quad g(\lambda, \theta) = \frac{1}{n}\sum_{i=1}^{n} g_i(\lambda, \theta)$$

Implicit differentiation in practice

Implicit gradient

$$\nabla \Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) - \nabla^2_{\lambda,\theta} g(\lambda, \theta^*(\lambda)) \boxed{\left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla_\theta f(\lambda, \theta^*(\lambda))}$$

Bottlenecks

- Solution of the inner problem
- Solution of a linear system
- Computing a gradient is expensive

ML setting: Empirical Risk Minimization

$$f(\lambda, \theta) = \frac{1}{m}\sum_{j=1}^{m} f_j(\lambda, \theta), \quad g(\lambda, \theta) = \frac{1}{n}\sum_{i=1}^{n} g_i(\lambda, \theta)$$

Stochastic optimization [Robbins & Monro '51]

$$\lambda^{t+1} = \lambda^t - \gamma^t d^t$$

Cheap estimator of $\nabla \Phi(\lambda^t)$

Implicit differentiation in practice

Implicit gradient

$$\nabla\Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) - \nabla^2_{\lambda,\theta} g(\lambda, \theta^*(\lambda)) \boxed{\left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla_\theta f(\lambda, \theta^*(\lambda))}$$

Bottlenecks

- Solution of the inner problem
- Solution of a linear system
- Computing a gradient is expensive

ML setting: Empirical Risk Minimization

$$f(\lambda, \theta) = \frac{1}{m}\sum_{j=1}^{m} f_j(\lambda, \theta), \quad g(\lambda, \theta) = \frac{1}{n}\sum_{i=1}^{n} g_i(\lambda, \theta)$$

Stochastic optimization [Robbins & Monro '51]

$$\lambda^{t+1} = \lambda^t - \gamma^t d^t$$

Cheap estimator of $\nabla\Phi(\lambda^t)$

Can we build an unbiased estimate of $\nabla\Phi(\lambda)$?

Implicit differentiation in practice

Implicit gradient

$$\nabla \Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) - \nabla^2_{\lambda,\theta} g(\lambda, \theta^*(\lambda)) \boxed{\left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla_\theta f(\lambda, \theta^*(\lambda))}$$

Bottlenecks

- Solution of the inner problem
- Solution of a linear system
- Computing a gradient is expensive

ML setting: Empirical Risk Minimization

$$f(\lambda, \theta) = \frac{1}{m} \sum_{j=1}^{m} f_j(\lambda, \theta), \quad g(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} g_i(\lambda, \theta)$$

Stochastic optimization[Robbins & Monro '51]

$$\lambda^{t+1} = \lambda^t - \gamma^t d^t$$

Cheap estimator of $\nabla \Phi(\lambda^t)$

Can we build an unbiased estimate of $\nabla \Phi(\lambda)$?

No straightforward answer since…

$$\left[\sum_{i=1}^{n} \nabla^2_{\theta,\theta} g_i\right]^{-1} \neq \sum_{i=1}^{n} \left[\nabla^2_{\theta,\theta} g_i\right]^{-1}$$

A framework for bilevel optimization that enables stochastic and global variance reduction algorithms

M. Dagréou, P. Ablin, S. Vaiter, T. Moreau. A framework for bilevel optimization that enables stochastic and global variance reduction algorithm. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. *Oral*

Framework for bilevel optimization

Implicit gradient

$$\nabla \Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) - \nabla^2_{\lambda,\theta} g(\lambda, \theta^*(\lambda)) \left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla_\theta f(\lambda, \theta^*(\lambda))$$

Framework for bilevel optimization

Implicit gradient

$$\nabla\Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) + \nabla^2_{\lambda,\theta} g(\lambda, \theta^*(\lambda)) v^*(\lambda)$$

$$v^*(\lambda) \triangleq -\left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla_\theta f(\lambda, \theta^*(\lambda))$$

Framework for bilevel optimization

Implicit gradient

$$\nabla\Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) + \nabla^2_{\lambda,\theta} g(\lambda, \theta^*(\lambda)) v^*(\lambda)$$

$$v^*(\lambda) \triangleq -\left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla_\theta f(\lambda, \theta^*(\lambda))$$

Main idea: Update θ, v and λ in the following directions:

Framework for bilevel optimization

Implicit gradient

$$\nabla\Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) + \nabla^2_{\lambda,\theta} g(\lambda, \theta^*(\lambda)) v^*(\lambda)$$

$$v^*(\lambda) \triangleq -\left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla_\theta f(\lambda, \theta^*(\lambda))$$

Main idea: Update θ, v and λ in the following directions:

- θ: $D_\theta(\theta, v, \lambda) = \nabla_\theta g(\lambda, \theta)$

Goes towards $\theta^*(\lambda)$

Framework for bilevel optimization

Implicit gradient

$$\nabla \Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) + \nabla^2_{\lambda,\theta} g(\lambda, \theta^*(\lambda)) v^*(\lambda)$$

$$v^*(\lambda) \triangleq - \left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla_\theta f(\lambda, \theta^*(\lambda))$$

Main idea: Update θ, v and λ in the following directions:

- θ: $D_\theta(\theta, v, \lambda) = \nabla_\theta g(\lambda, \theta)$ — *Goes towards* $\theta^*(\lambda)$
- v: $D_v(\theta, v, \lambda) = \nabla^2_{\theta,\theta} g(\lambda, \theta) v + \nabla_\theta f(\lambda, \theta)$ — *Goes towards* $v^*(\lambda)$

Framework for bilevel optimization

Implicit gradient

$$\nabla\Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) + \nabla^2_{\lambda,\theta} g(\lambda, \theta^*(\lambda)) v^*(\lambda)$$

$$v^*(\lambda) \triangleq -\left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla_\theta f(\lambda, \theta^*(\lambda))$$

Main idea: Update θ, v and λ in the following directions:

- θ: $D_\theta(\theta, v, \lambda) = \nabla_\theta g(\lambda, \theta)$ — *Goes towards $\theta^*(\lambda)$*
- v: $D_v(\theta, v, \lambda) = \nabla^2_{\theta,\theta} g(\lambda, \theta) v + \nabla_\theta f(\lambda, \theta)$ — *Goes towards $v^*(\lambda)$*
- λ: $D_\lambda(\theta, v, \lambda) = \nabla^2_{\lambda,\theta} g(\lambda, \theta) v + \nabla_\lambda f(\lambda, \theta)$ — *Approximate gradient step*

Framework for bilevel optimization

Implicit gradient

$$\nabla\Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) + \nabla^2_{\lambda,\theta} g(\lambda, \theta^*(\lambda)) v^*(\lambda)$$

$$v^*(\lambda) \triangleq -\left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla_\theta f(\lambda, \theta^*(\lambda))$$

Main idea: Update θ, v and λ in the following directions:

- θ: $D_\theta(\theta, v, \lambda) = \nabla_\theta g(\lambda, \theta)$ — *Goes towards $\theta^*(\lambda)$*
- v: $D_v(\theta, v, \lambda) = \nabla^2_{\theta,\theta} g(\lambda, \theta) v + \nabla_\theta f(\lambda, \theta)$ — *Goes towards $v^*(\lambda)$*
- λ: $D_\lambda(\theta, v, \lambda) = \nabla^2_{\lambda,\theta} g(\lambda, \theta) v + \nabla_\lambda f(\lambda, \theta)$ — *Approximate gradient step*

$$\longrightarrow D_\lambda(\theta^*(\lambda), v^*(\lambda), \lambda) = \nabla\Phi(\lambda)$$

Framework for bilevel optimization

Implicit gradient

$$\nabla\Phi(\lambda) = \nabla_\lambda f(\lambda, \theta^*(\lambda)) + \nabla^2_{\lambda,\theta} g(\lambda, \theta^*(\lambda)) v^*(\lambda)$$

$$v^*(\lambda) \triangleq -\left[\nabla^2_{\theta,\theta} g(\lambda, \theta^*(\lambda))\right]^{-1} \nabla_\theta f(\lambda, \theta^*(\lambda))$$

Main idea: Update θ, v and λ in the following directions:

- θ: $D_\theta(\theta, v, \lambda) = \nabla_\theta g(\lambda, \theta)$ — *Goes towards $\theta^*(\lambda)$*
- v: $D_v(\theta, v, \lambda) = \nabla^2_{\theta,\theta} g(\lambda, \theta) v + \nabla_\theta f(\lambda, \theta)$ — *Goes towards $v^*(\lambda)$*
- λ: $D_\lambda(\theta, v, \lambda) = \nabla^2_{\lambda,\theta} g(\lambda, \theta) v + \nabla_\lambda f(\lambda, \theta)$ — *Approximate gradient step*

$$\rightarrow \quad D_\lambda(\theta^*(\lambda), v^*(\lambda), \lambda) = \nabla\Phi(\lambda)$$

Bilevel Dynamics

$$\begin{bmatrix} \theta^{t+1} \\ v^{t+1} \\ \lambda^{t+1} \end{bmatrix} = \begin{bmatrix} \theta^t - \rho^t D_\theta(\theta^t, v^t, \lambda^t) \\ v^t - \rho^t D_v(\theta^t, v^t, \lambda^t) \\ \lambda^t - \gamma^t D_\lambda(\theta^t, v^t, \lambda^t) \end{bmatrix}$$

Same step size in θ and v because same conditioning

Framework for *stochastic* bilevel optimization

Update directions

- $D_\theta(\theta, v, \lambda) = \nabla_\theta g(\lambda, \theta)$
- $D_v(\theta, v, \lambda) = \nabla^2_{\theta,\theta} g(\lambda, \theta) v + \nabla_\theta f(\lambda, \theta)$
- $D_\lambda(\theta, v, \lambda) = \nabla^2_{\lambda,\theta} g(\lambda, \theta) v + \nabla_\lambda f(\lambda, \theta)$

Framework for *stochastic* bilevel optimization

Update directions

- $D_\theta(\theta, v, \lambda) = \nabla_\theta g(\lambda, \theta)$
- $D_v(\theta, v, \lambda) = \nabla^2_{\theta,\theta} g(\lambda, \theta) v + \nabla_\theta f(\lambda, \theta)$
- $D_\lambda(\theta, v, \lambda) = \nabla^2_{\lambda,\theta} g(\lambda, \theta) v + \nabla_\lambda f(\lambda, \theta)$

ERM

$$f(\lambda, \theta) = \frac{1}{m} \sum_{j=1}^{m} f_j(\lambda, \theta), \quad g(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} g_i(\lambda, \theta)$$

Framework for *stochastic* bilevel optimization

Update directions

- $D_\theta(\theta, v, \lambda) = \nabla_\theta g(\lambda, \theta)$
- $D_v(\theta, v, \lambda) = \nabla^2_{\theta,\theta} g(\lambda, \theta) v + \nabla_\theta f(\lambda, \theta)$
- $D_\lambda(\theta, v, \lambda) = \nabla^2_{\lambda,\theta} g(\lambda, \theta) v + \nabla_\lambda f(\lambda, \theta)$

Linear in f and g

ERM

$$f(\lambda, \theta) = \frac{1}{m} \sum_{j=1}^{m} f_j(\lambda, \theta), \quad g(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} g_i(\lambda, \theta)$$

Framework for *stochastic* bilevel optimization

Update directions

- $D_\theta(\theta, v, \lambda) = \nabla_\theta g(\lambda, \theta)$
- $D_v(\theta, v, \lambda) = \nabla^2_{\theta,\theta} g(\lambda, \theta) v + \nabla_\theta f(\lambda, \theta)$
- $D_\lambda(\theta, v, \lambda) = \nabla^2_{\lambda,\theta} g(\lambda, \theta) v + \nabla_\lambda f(\lambda, \theta)$

Linear in f and g

Stochastic Bilevel Dynamics

$$\begin{bmatrix} \theta^{t+1} \\ v^{t+1} \\ \lambda^{t+1} \end{bmatrix} = \begin{bmatrix} \theta^t - \rho^t D_\theta^t \\ v^t - \rho^t D_v^t \\ \lambda^t - \gamma^t D_\lambda^t \end{bmatrix}$$

ERM

$$f(\lambda, \theta) = \frac{1}{m} \sum_{j=1}^{m} f_j(\lambda, \theta), \quad g(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} g_i(\lambda, \theta)$$

Framework for *stochastic* bilevel optimization

Update directions

- $D_\theta(\theta, v, \lambda) = \nabla_\theta g(\lambda, \theta)$
- $D_v(\theta, v, \lambda) = \nabla^2_{\theta,\theta} g(\lambda, \theta) v + \nabla_\theta f(\lambda, \theta)$
- $D_\lambda(\theta, v, \lambda) = \nabla^2_{\lambda,\theta} g(\lambda, \theta) v + \nabla_\lambda f(\lambda, \theta)$

Linear in f and g

ERM

$$f(\lambda, \theta) = \frac{1}{m} \sum_{j=1}^{m} f_j(\lambda, \theta), \quad g(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} g_i(\lambda, \theta)$$

Stochastic Bilevel Dynamics

$$\begin{bmatrix} \theta^{t+1} \\ v^{t+1} \\ \lambda^{t+1} \end{bmatrix} = \begin{bmatrix} \theta^t - \rho^t D_\theta^t \\ v^t - \rho^t D_v^t \\ \lambda^t - \gamma^t D_\lambda^t \end{bmatrix}$$

Stochastic estimators of $D_\theta(\theta^t, v^t, \lambda^t)$, $D_v(\theta^t, v^t, \lambda^t)$ and $D_\lambda(\theta^t, v^t, \lambda^t)$

Framework for *stochastic* bilevel optimization

Update directions

- $D_\theta(\theta, v, \lambda) = \nabla_\theta g(\lambda, \theta)$
- $D_v(\theta, v, \lambda) = \nabla^2_{\theta,\theta} g(\lambda, \theta) v + \nabla_\theta f(\lambda, \theta)$
- $D_\lambda(\theta, v, \lambda) = \nabla^2_{\lambda,\theta} g(\lambda, \theta) v + \nabla_\lambda f(\lambda, \theta)$

Linear in f and g

Stochastic Bilevel Dynamics

$$\begin{bmatrix} \theta^{t+1} \\ v^{t+1} \\ \lambda^{t+1} \end{bmatrix} = \begin{bmatrix} \theta^t - \rho^t D^t_\theta \\ v^t - \rho^t D^t_v \\ \lambda^t - \gamma^t D^t_\lambda \end{bmatrix}$$

Stochastic estimators of $D_\theta(\theta^t, v^t, \lambda^t)$, $D_v(\theta^t, v^t, \lambda^t)$ and $D_\lambda(\theta^t, v^t, \lambda^t)$

ERM

$$f(\lambda, \theta) = \frac{1}{m} \sum_{j=1}^{m} f_j(\lambda, \theta), \quad g(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} g_i(\lambda, \theta)$$

SOBA directions

Sample $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$and set

$$D^t_\theta = \nabla_\theta g_i(\lambda^t, \theta^t)$$
$$D^t_v = \nabla^2_{\theta,\theta} g_i(\lambda^t, \theta^t) v^t + \nabla_\theta f_j(\lambda^t, \theta^t)$$
$$D^t_\lambda = \nabla^2_{\lambda,\theta} g_i(\lambda^t, \theta^t) v^t + \nabla_\lambda f_j(\lambda^t, \theta^t)$$

SOBA (StOchastic Bilevel Algorithm) in details

SOBA directions

Sample $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$ and set

$$D_\theta^t = \nabla_\theta g_i(\lambda^t, \theta^t)$$
$$D_v^t = \nabla^2_{\theta,\theta} g_i(\lambda^t, \theta^t) v^t + \nabla_\theta f_j(\lambda^t, \theta^t)$$
$$D_\lambda^t = \nabla^2_{\lambda,\theta} g_i(\lambda^t, \theta^t) v^t + \nabla_\lambda f_j(\lambda^t, \theta^t)$$

SOBA (StOchastic Bilevel Algorithm) in details

SOBA directions

Sample $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$ and set

$$D_\theta^t = \nabla_\theta g_i(\lambda^t, \theta^t)$$

$$D_v^t = \nabla^2_{\theta,\theta} g_i(\lambda^t, \theta^t) v^t + \nabla_\theta f_j(\lambda^t, \theta^t)$$

$$D_\lambda^t = \nabla^2_{\lambda,\theta} g_i(\lambda^t, \theta^t) v^t + \nabla_\lambda f_j(\lambda^t, \theta^t)$$

Iteration cost

SOBA (StOchastic Bilevel Algorithm) in details

SOBA directions

Sample $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ and set

$$D_\theta^t = \nabla_\theta g_i(\lambda^t, \theta^t)$$

$$D_v^t = \nabla^2_{\theta,\theta} g_i(\lambda^t, \theta^t) v^t + \nabla_\theta f_j(\lambda^t, \theta^t)$$

$$D_\lambda^t = \nabla^2_{\lambda,\theta} g_i(\lambda^t, \theta^t) v^t + \nabla_\lambda f_j(\lambda^t, \theta^t)$$

Iteration cost

- Gradients: computed efficiently by reverse mode automatic differentiation [Linnainmaa et al. '70]

SOBA (StOchastic Bilevel Algorithm) in details

SOBA directions

Sample $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$ and set

$$D_\theta^t = \nabla_\theta g_i(\lambda^t, \theta^t)$$
$$D_v^t = \boxed{\nabla^2_{\theta,\theta} g_i(\lambda^t, \theta^t) v^t} + \nabla_\theta f_j(\lambda^t, \theta^t)$$
$$D_\lambda^t = \boxed{\nabla^2_{\lambda,\theta} g_i(\lambda^t, \theta^t) v^t} + \nabla_\lambda f_j(\lambda^t, \theta^t)$$

Iteration cost

- Gradients: computed efficiently by reverse mode automatic differentiation [Linnainmaa et al. '70]
- HVPs: At first sight 😱 😱 😱 😱, but…

How to compute Hessian-vector products?

M. Dagréou, P. Ablin, S. Vaiter, T. Moreau. How to compute Hessian-vector products? In *ICLR blogpost track*, 2024. *Spotlight*

Efficient computation by automatic differentiation [Pearlmutter ’94]

How to compute Hessian-vector products?

M. Dagréou, P. Ablin, S. Vaiter, T. Moreau. How to compute Hessian-vector products? In *ICLR blogpost track*, 2024. *Spotlight*

Efficient computation by automatic differentiation [Pearlmutter '94]

$$\nabla^2_{\theta,\theta} g_i(\lambda^t, \theta^t) v^t = \nabla_\theta \left[\langle \nabla_\theta g_i(\lambda^t, \theta^t), v^t \rangle\right]$$

How to compute Hessian-vector products?

M. Dagréou, P. Ablin, S. Vaiter, T. Moreau. How to compute Hessian-vector products? In *ICLR blogpost track*, 2024. *Spotlight*

Efficient computation by automatic differentiation [Pearlmutter '94]

$$\nabla^2_{\theta,\theta} g_i(\lambda^t, \theta^t) v^t = \nabla_\theta \left[\langle \nabla_\theta g_i(\lambda^t, \theta^t), v^t \rangle \right]$$

- reverse-over-reverse: « grad of the JVP »

```
jax.grad(lambda y: jnp.vdot(jax.grad(g)(y), v))(params)
```

How to compute Hessian-vector products?

M. Dagréou, P. Ablin, S. Vaiter, T. Moreau. How to compute Hessian-vector products? In *ICLR blogpost track*, 2024. *Spotlight*

Efficient computation by automatic differentiation [Pearlmutter '94]

$$\nabla^2_{\theta,\theta} g_i(\lambda^t, \theta^t) v^t = \nabla_\theta \left[\langle \nabla_\theta g_i(\lambda^t, \theta^t), v^t \rangle \right]$$

- reverse-over-reverse: « grad of the JVP »

```
jax.grad(lambda y: jnp.vdot(jax.grad(g)(y), v))(params)
```

- reverse-over-forward: « grad of the JVP »

```
jax.grad(lambda y: jax.jvp(g, (y, ), (v, ))[1])(params)
```

How to compute Hessian-vector products?

M. Dagréou, P. Ablin, S. Vaiter, T. Moreau. How to compute Hessian-vector products? In *ICLR blogpost track*, 2024. *Spotlight*

Efficient computation by automatic differentiation [Pearlmutter '94]

$$\nabla^2_{\theta,\theta} g_i(\lambda^t, \theta^t) v^t = \nabla_\theta \left[\langle \nabla_\theta g_i(\lambda^t, \theta^t), v^t \rangle \right]$$

- reverse-over-reverse: « grad of the JVP »

```
jax.grad(lambda y: jnp.vdot(jax.grad(g)(y), v))(params)
```

- reverse-over-forward: « grad of the JVP »

```
jax.grad(lambda y: jax.jvp(g, (y, ), (v, ))[1])(params)
```

- forward-over-reverse: « JVP of the grad »

```
jax.jvp(jax.grad(g), (params, ), (v, ))[1]
```

How to compute Hessian-vector products?

M. Dagréou, P. Ablin, S. Vaiter, T. Moreau. How to compute Hessian-vector products? In *ICLR blogpost track*, 2024. *Spotlight*

Efficient computation by automatic differentiation [Pearlmutter '94]

$$\nabla^2_{\theta,\theta} g_i(\lambda^t, \theta^t) v^t = \nabla_\theta \left[\langle \nabla_\theta g_i(\lambda^t, \theta^t), v^t \rangle \right]$$

- reverse-over-reverse: « grad of the JVP »

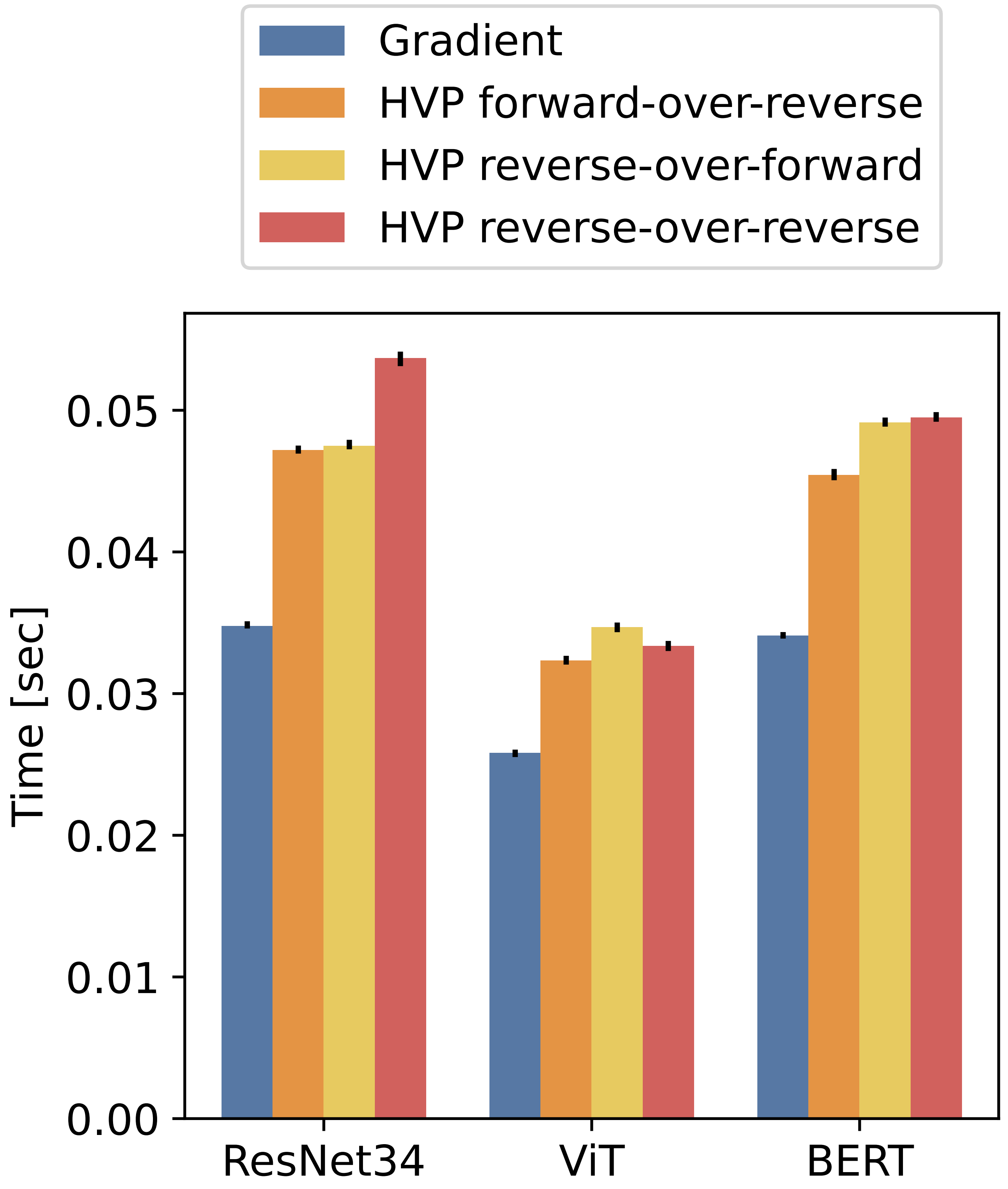
```
jax.grad(lambda y: jnp.vdot(jax.grad(g)(y), v))(params)
```

- reverse-over-forward: « grad of the JVP »

```
jax.grad(lambda y: jax.jvp(g, (y, ), (v, ))[1])(params)
```

- forward-over-reverse: « JVP of the grad »

```
jax.jvp(jax.grad(g), (params, ), (v, ))[1]
```



How to compute Hessian-vector products?

M. Dagréou, P. Ablin, S. Vaiter, T. Moreau. How to compute Hessian-vector products? In *ICLR blogpost track*, 2024. *Spotlight*

Efficient computation by automatic differentiation [Pearlmutter '94]

$$\nabla^2_{\theta,\theta} g_i(\lambda^t, \theta^t) v^t = \nabla_\theta \left[\langle \nabla_\theta g_i(\lambda^t, \theta^t), v^t \rangle \right]$$

- reverse-over-reverse: « grad of the JVP »

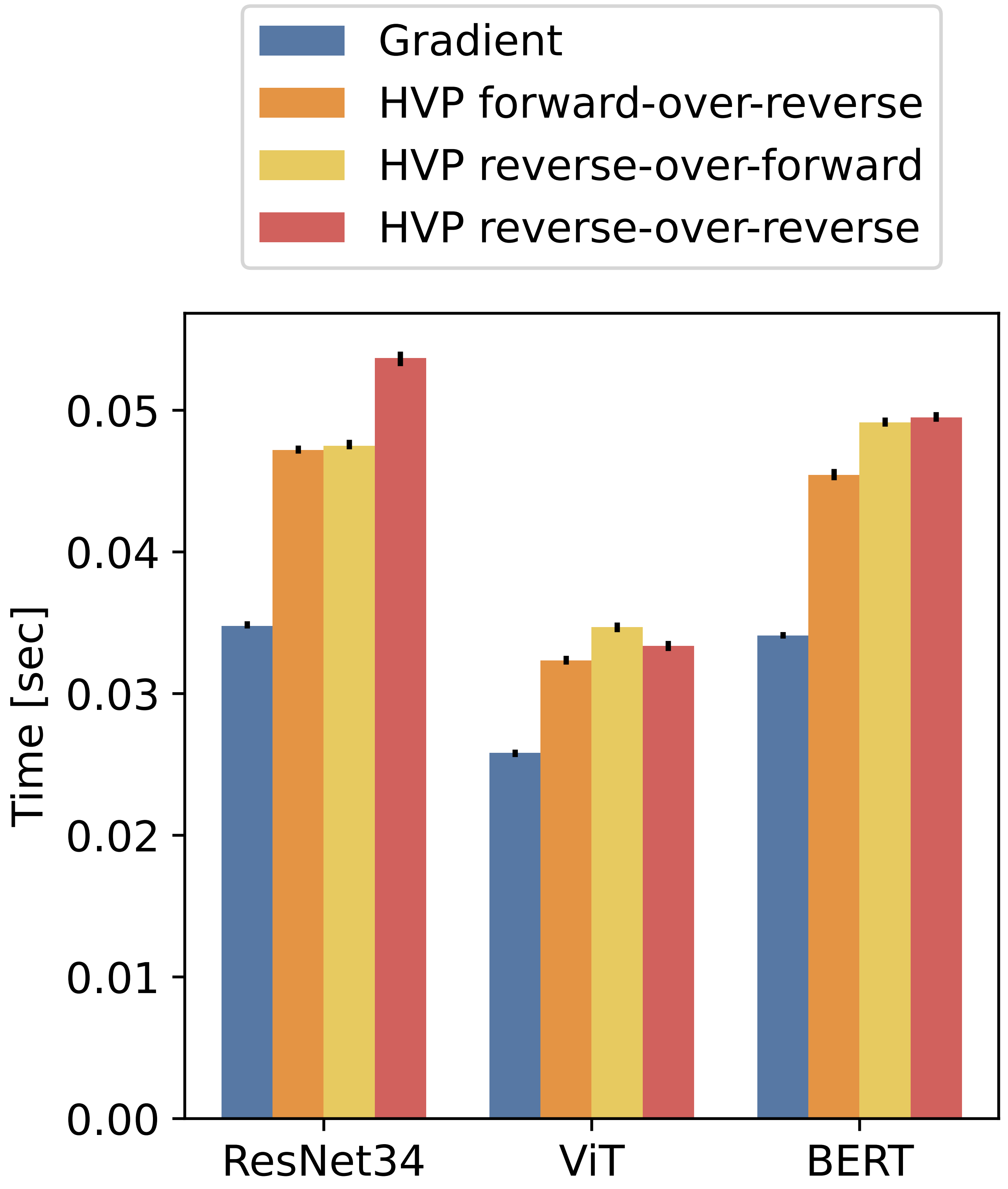
```
jax.grad(lambda y: jnp.vdot(jax.grad(g)(y), v))(params)
```

- reverse-over-forward: « grad of the JVP »

```
jax.grad(lambda y: jax.jvp(g, (y, ), (v, ))[1])(params)
```

- forward-over-reverse: « JVP of the grad »

```
jax.jvp(jax.grad(g), (params, ), (v, ))[1]
```



The HVP cost scales as the gradient cost!!!

Convergence of SOBA

Convergence of SOBA

Theorem

Assume that

Convergence of SOBA

Theorem

Assume that

1. the outer function f is twice differentiable with Lipschitz derivatives

Convergence of SOBA

Theorem

Assume that

1. the outer function f is twice differentiable with Lipschitz derivatives
2. the inner function g is three times differentiable with Lipschitz derivatives

Convergence of SOBA

Theorem

Assume that

1. the outer function f is twice differentiable with Lipschitz derivatives
2. the inner function g is three times differentiable with Lipschitz derivatives
3. the stochastic directions verify

$$\mathbb{E}_t\left[\|D_\theta^t\|^2\right] \leq B_\theta(1 + \|D_\theta(\theta^t, v^t, \lambda^t)\|^2), \quad \mathbb{E}_t\left[\|D_v^t\|^2\right] \leq B_v(1 + \|D_v(\theta^t, v^t, \lambda^t)\|^2),$$

$$\mathbb{E}_t\left[\|D_\lambda^t\|^2\right] \leq B_\lambda$$

Convergence of SOBA

Theorem

Assume that

1. the outer function f is twice differentiable with Lipschitz derivatives
2. the inner function g is three times differentiable with Lipschitz derivatives
3. the stochastic directions verify

$$\mathbb{E}_t\left[\|D_\theta^t\|^2\right] \leq B_\theta(1 + \|D_\theta(\theta^t, v^t, \lambda^t)\|^2), \quad \mathbb{E}_t\left[\|D_v^t\|^2\right] \leq B_v(1 + \|D_v(\theta^t, v^t, \lambda^t)\|^2),$$

$$\mathbb{E}_t\left[\|D_\lambda^t\|^2\right] \leq B_\lambda$$

Then, for step sizes $\rho^t \asymp t^{-\frac{1}{2}}$ and $\gamma^t \asymp t^{-\frac{1}{2}}$ it holds

$$\inf_{0 \leq t \leq T-1} \mathbb{E}[\|\nabla \Phi(\lambda^t)\|^2] \leq \mathcal{O}(\log(T) T^{-\frac{1}{2}})$$

Convergence of SOBA

Theorem

Assume that

1. the outer function f is twice differentiable with Lipschitz derivatives
2. the inner function g is three times differentiable with Lipschitz derivatives
3. the stochastic directions verify

$$\mathbb{E}_t\left[\|D_\theta^t\|^2\right] \leq B_\theta(1 + \|D_\theta(\theta^t, v^t, \lambda^t)\|^2), \quad \mathbb{E}_t\left[\|D_v^t\|^2\right] \leq B_v(1 + \|D_v(\theta^t, v^t, \lambda^t)\|^2),$$

$$\mathbb{E}_t\left[\|D_\lambda^t\|^2\right] \leq B_\lambda$$

Then, for step sizes $\rho^t \asymp t^{-\frac{1}{2}}$ and $\gamma^t \asymp t^{-\frac{1}{2}}$ it holds

Decreasing step sizes

$$\inf_{0 \leq t \leq T-1} \mathbb{E}[\|\nabla\Phi(\lambda^t)\|^2] \leq \mathcal{O}(\log(T)T^{-\frac{1}{2}})$$

Convergence of SOBA

Theorem

Assume that

1. the outer function f is twice differentiable with Lipschitz derivatives
2. the inner function g is three times differentiable with Lipschitz derivatives
3. the stochastic directions verify

$$\mathbb{E}_t\left[\|D_\theta^t\|^2\right] \leq B_\theta(1 + \|D_\theta(\theta^t, v^t, \lambda^t)\|^2), \quad \mathbb{E}_t\left[\|D_v^t\|^2\right] \leq B_v(1 + \|D_v(\theta^t, v^t, \lambda^t)\|^2),$$

$$\mathbb{E}_t\left[\|D_\lambda^t\|^2\right] \leq B_\lambda$$

Then, for step sizes $\rho^t \asymp t^{-\frac{1}{2}}$ and $\gamma^t \asymp t^{-\frac{1}{2}}$ it holds

$$\inf_{0\leq t\leq T-1} \mathbb{E}[\|\nabla\Phi(\lambda^t)\|^2] \leq \mathcal{O}(\log(T)T^{-\frac{1}{2}})$$

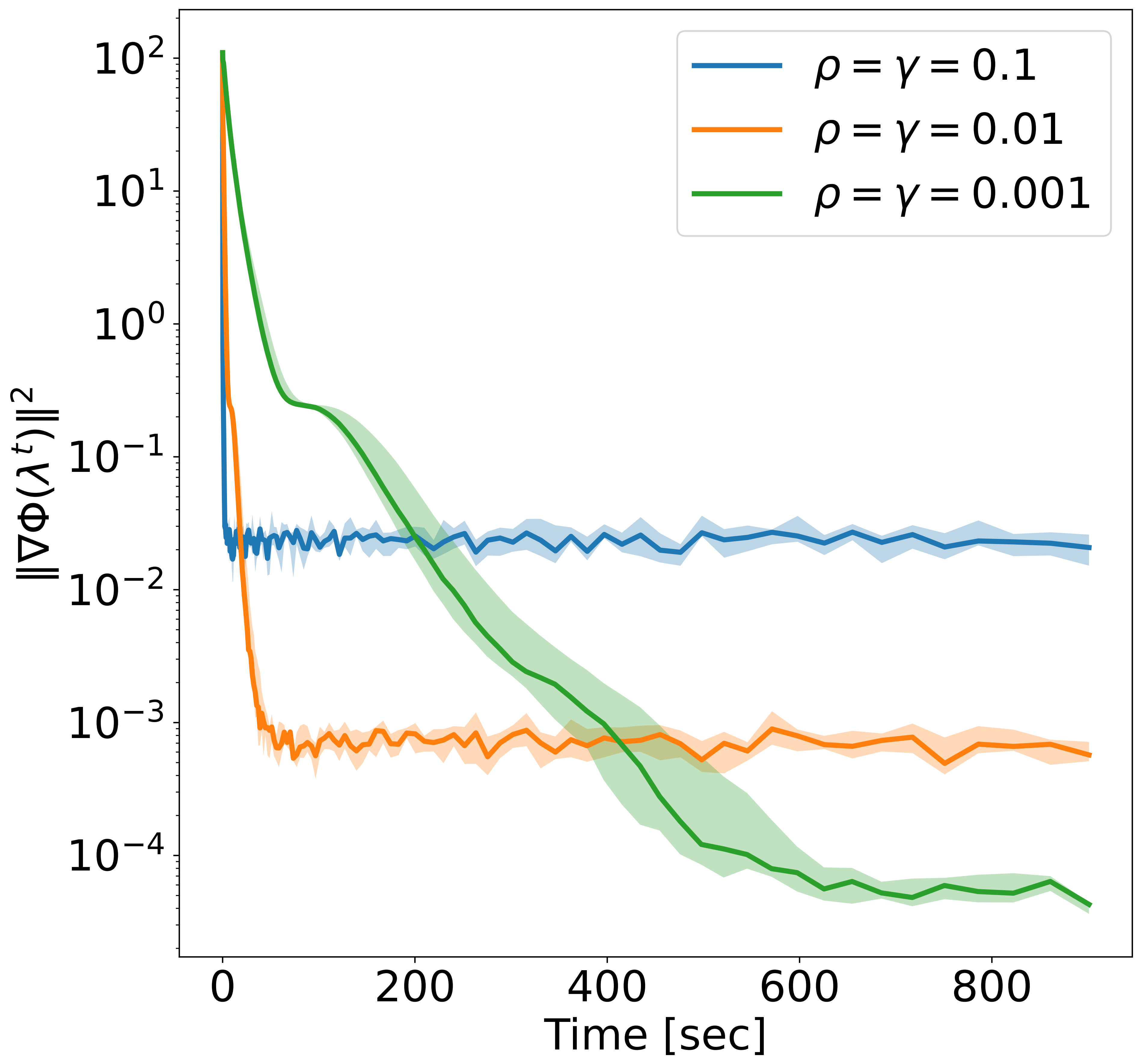
Similar to the rate of SGD for non-convex smooth functions [Ghadimi '13]

Why decreasing the step sizes?

Quadratic setting

$$f(\lambda, \theta) = \frac{1}{m} \sum_{j=1}^{m} \left\langle A_j^f \begin{bmatrix} \lambda \\ \theta \end{bmatrix}, \begin{bmatrix} \lambda \\ \theta \end{bmatrix} \right\rangle + \left\langle b_j^f, \begin{bmatrix} \lambda \\ \theta \end{bmatrix} \right\rangle + c_j^f$$

$$g(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} \left\langle A_i^g \begin{bmatrix} \lambda \\ \theta \end{bmatrix}, \begin{bmatrix} \lambda \\ \theta \end{bmatrix} \right\rangle + \left\langle b_i^g, \begin{bmatrix} \lambda \\ \theta \end{bmatrix} \right\rangle + c_i^g$$



$$\delta_\theta^t = \mathbb{E}\left[\|\theta^t - \theta^*(\lambda^t)\|^2\right]$$

Why decreasing the step sizes?

$$\delta_v^t = \mathbb{E}\left[\|v^t - v^*(\lambda^t)\|^2\right]$$

$$\Phi^t = \mathbb{E}[\Phi(\lambda^t)]$$

Fundamental descent lemma

$$\delta_\theta^{t+1} \leq (1-\rho\mu_g)\delta_\theta^t + \rho^2\mathbb{E}[\|D_\theta^t\|^2] + \rho^2\mathbb{E}[\|D_\lambda^t\|^2] + \frac{\gamma^2}{\rho}\mathbb{E}[\|D_\lambda(\theta^t, v^t, \lambda^t)\|^2]$$

$$\delta_v^{t+1} \leq (1-\rho\mu_g)\delta_v^t + \rho\delta_\theta^t + \rho^2\mathbb{E}[\|D_v^t\|^2] + \rho^2\mathbb{E}[\|D_\lambda^t\|^2] + \frac{\gamma^2}{\rho}\mathbb{E}[\|D_\lambda(\theta^t, v^t, \lambda^t)\|^2]$$

$$\Phi^{t+1} \leq \Phi^t - \gamma\mathbb{E}\left[\|\nabla\Phi(\lambda^t)\|^2\right] - \gamma\mathbb{E}[\|D_\lambda(\theta^t, v^t, \lambda^t)\|] + \gamma(\delta_\theta^t + \delta_v^t) + \gamma^2\mathbb{E}\left[\|D_\lambda^t\|^2\right]$$

Why decreasing the step sizes?

$$\delta_\theta^t = \mathbb{E}\left[\|\theta^t - \theta^*(\lambda^t)\|^2\right]$$

$$\delta_v^t = \mathbb{E}\left[\|v^t - v^*(\lambda^t)\|^2\right]$$

$$\Phi^t = \mathbb{E}[\Phi(\lambda^t)]$$

Fundamental descent lemma

$$\delta_\theta^{t+1} \leq (1-\rho\mu_g)\delta_\theta^t + \rho^2\mathbb{E}[\|D_\theta^t\|^2] + \rho^2\mathbb{E}[\|D_\lambda^t\|^2] + \frac{\gamma^2}{\rho}\mathbb{E}[\|D_\lambda(\theta^t, v^t, \lambda^t)\|^2]$$

$$\delta_v^{t+1} \leq (1-\rho\mu_g)\delta_v^t + \rho\delta_\theta^t + \rho^2\mathbb{E}[\|D_v^t\|^2] + \rho^2\mathbb{E}[\|D_\lambda^t\|^2] + \frac{\gamma^2}{\rho}\mathbb{E}[\|D_\lambda(\theta^t, v^t, \lambda^t)\|^2]$$

$$\Phi^{t+1} \leq \Phi^t - \gamma\mathbb{E}\left[\|\nabla\Phi(\lambda^t)\|^2\right] - \gamma\mathbb{E}[\|D_\lambda(\theta^t, v^t, \lambda^t)\|] + \gamma(\delta_\theta^t + \delta_v^t) + \gamma^2\mathbb{E}\left[\|D_\lambda^t\|^2\right]$$

Variance terms prevent from converging if not converging towards 0

$$\delta_\theta^t = \mathbb{E}\left[\|\theta^t - \theta^*(\lambda^t)\|^2\right]$$

$$\delta_v^t = \mathbb{E}\left[\|v^t - v^*(\lambda^t)\|^2\right]$$

$$\Phi^t = \mathbb{E}[\Phi(\lambda^t)]$$

Why decreasing the step sizes?

Fundamental descent lemma

$$\delta_\theta^{t+1} \leq (1-\rho\mu_g)\delta_\theta^t + \boxed{\rho^2\mathbb{E}[\|D_\theta^t\|^2]} + \boxed{\rho^2\mathbb{E}[\|D_\lambda^t\|^2]} + \frac{\gamma^2}{\rho}\mathbb{E}[\|D_\lambda(\theta^t, v^t, \lambda^t)\|^2]$$

$$\delta_v^{t+1} \leq (1-\rho\mu_g)\delta_v^t + \rho\delta_\theta^t + \boxed{\rho^2\mathbb{E}[\|D_v^t\|^2]} + \boxed{\rho^2\mathbb{E}[\|D_\lambda^t\|^2]} + \frac{\gamma^2}{\rho}\mathbb{E}[\|D_\lambda(\theta^t, v^t, \lambda^t)\|^2]$$

$$\Phi^{t+1} \leq \Phi^t - \gamma\mathbb{E}\left[\|\nabla\Phi(\lambda^t)\|^2\right] - \gamma\mathbb{E}[\|D_\lambda(\theta^t, v^t, \lambda^t)\|] + \gamma(\delta_\theta^t + \delta_v^t) + \boxed{\gamma^2\mathbb{E}\left[\|D_\lambda^t\|^2\right]}$$

Variance terms prevent from converging if not converging towards 0

- Make the step sizes decreasing -> leads to slow convergence

$$\delta_\theta^t = \mathbb{E}\left[\|\theta^t - \theta^*(\lambda^t)\|^2\right]$$

$$\delta_v^t = \mathbb{E}\left[\|v^t - v^*(\lambda^t)\|^2\right]$$

$$\Phi^t = \mathbb{E}[\Phi(\lambda^t)]$$

Why decreasing the step sizes?

Fundamental descent lemma

$$\delta_\theta^{t+1} \leq (1-\rho\mu_g)\delta_\theta^t + \boxed{\rho^2\mathbb{E}[\|D_\theta^t\|^2]} + \boxed{\rho^2\mathbb{E}[\|D_\lambda^t\|^2]} + \frac{\gamma^2}{\rho}\mathbb{E}[\|D_\lambda(\theta^t, v^t, \lambda^t)\|^2]$$

$$\delta_v^{t+1} \leq (1-\rho\mu_g)\delta_v^t + \rho\delta_\theta^t + \boxed{\rho^2\mathbb{E}[\|D_v^t\|^2]} + \boxed{\rho^2\mathbb{E}[\|D_\lambda^t\|^2]} + \frac{\gamma^2}{\rho}\mathbb{E}[\|D_\lambda(\theta^t, v^t, \lambda^t)\|^2]$$

$$\Phi^{t+1} \leq \Phi^t - \gamma\mathbb{E}\left[\|\nabla\Phi(\lambda^t)\|^2\right] - \gamma\mathbb{E}[\|D_\lambda(\theta^t, v^t, \lambda^t)\|] + \gamma(\delta_\theta^t + \delta_v^t) + \boxed{\gamma^2\mathbb{E}\left[\|D_\lambda^t\|^2\right]}$$

Variance terms prevent from converging if not converging towards 0

- Make the step sizes decreasing -> leads to slow convergence
- Make the variance decrease? -> Variance reduction algorithms[Johnson et al. 13, Defazio et al. ’14, Bietti & Mairal ‘17]

SRBA (Stochastic Recursive Bilevel Algorithm)

SRBA (Stochastic Recursive Bilevel Algorithm)

General principle

- Adaptation of SARAH/SPIDER to bilevel setting [Nguyen et al. ’17, Fang et al. ’18]

SRBA (Stochastic Recursive Bilevel Algorithm)

General principle

- Adaptation of SARAH/SPIDER to bilevel setting [Nguyen et al. ’17, Fang et al. ’18]
- Recursive estimate of the directions

Recursive estimation of the directions

Sample $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$ and set

$$
\begin{aligned}
D_\theta^{t,k} &= D_\theta^{i,j}(\theta^{t,k}, v^{t,k}, \lambda^{t,k}) - D_\theta^{i,j}(\theta^{t,k-1}, v^{t,k-1}, \lambda^{t,k-1}) + D_\theta^{t,k-1} \\
D_v^{t,k} &= D_v^{i,j}(\theta^{t,k}, v^{t,k}, \lambda^{t,k}) - D_v^{i,j}(\theta^{t,k-1}, v^{t,k-1}, \lambda^{t,k-1}) + D_v^{t,k-1} \\
D_\lambda^{t,k} &= D_\lambda^{i,j}(\theta^{t,k}, v^{t,k}, \lambda^{t,k}) - D_\lambda^{i,j}(\theta^{t,k-1}, v^{t,k-1}, \lambda^{t,k-1}) + D_\lambda^{t,k-1}
\end{aligned}
$$

SRBA (Stochastic Recursive Bilevel Algorithm)

General principle

- Adaptation of SARAH/SPIDER to bilevel setting [Nguyen et al. '17, Fang et al. '18]
- Recursive estimate of the directions

Recursive estimation of the directions

Sample $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$ and set

$$D_\theta^{t,k} = D_\theta^{i,j}(\theta^{t,k}, v^{t,k}, \lambda^{t,k}) - D_\theta^{i,j}(\theta^{t,k-1}, v^{t,k-1}, \lambda^{t,k-1}) + D_\theta^{t,k-1}$$

$$D_v^{t,k} = D_v^{i,j}(\theta^{t,k}, v^{t,k}, \lambda^{t,k}) - D_v^{i,j}(\theta^{t,k-1}, v^{t,k-1}, \lambda^{t,k-1}) + D_v^{t,k-1}$$

$$D_\lambda^{t,k} = D_\lambda^{i,j}(\theta^{t,k}, v^{t,k}, \lambda^{t,k}) - D_\lambda^{i,j}(\theta^{t,k-1}, v^{t,k-1}, \lambda^{t,k-1}) + D_\lambda^{t,k-1}$$

Inner loop index

SRBA (Stochastic Recursive Bilevel Algorithm)

General principle

- Adaptation of SARAH/SPIDER to bilevel setting [Nguyen et al. ’17, Fang et al. ’18]
- Recursive estimate of the directions

Recursive estimation of the directions

Sample $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ and set

$$D_\theta^{t,k} = D_\theta^{i,j}(\theta^{t,k}, v^{t,k}, \lambda^{t,k}) - D_\theta^{i,j}(\theta^{t,k-1}, v^{t,k-1}, \lambda^{t,k-1}) + D_\theta^{t,k-1}$$

$$D_v^{t,k} = D_v^{i,j}(\theta^{t,k}, v^{t,k}, \lambda^{t,k}) - D_v^{i,j}(\theta^{t,k-1}, v^{t,k-1}, \lambda^{t,k-1}) + D_v^{t,k-1}$$

$$D_\lambda^{t,k} = D_\lambda^{i,j}(\theta^{t,k}, v^{t,k}, \lambda^{t,k}) - D_\lambda^{i,j}(\theta^{t,k-1}, v^{t,k-1}, \lambda^{t,k-1}) + D_\lambda^{t,k-1}$$

Outer loop index

Inner loop index

SRBA (Stochastic Recursive Bilevel Algorithm)

General principle

- Adaptation of SARAH/SPIDER to bilevel setting [Nguyen et al. '17, Fang et al. '18]
- Recursive estimate of the directions

Recursive estimation of the directions

Sample $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ and set

$$D_\theta^{t,k} = D_\theta^{i,j}(\theta^{t,k}, v^{t,k}, \lambda^{t,k}) - D_\theta^{i,j}(\theta^{t,k-1}, v^{t,k-1}, \lambda^{t,k-1}) + D_\theta^{t,k-1}$$

$$D_v^{t,k} = D_v^{i,j}(\theta^{t,k}, v^{t,k}, \lambda^{t,k}) - D_v^{i,j}(\theta^{t,k-1}, v^{t,k-1}, \lambda^{t,k-1}) + D_v^{t,k-1}$$

$$D_\lambda^{t,k} = D_\lambda^{i,j}(\theta^{t,k}, v^{t,k}, \lambda^{t,k}) - D_\lambda^{i,j}(\theta^{t,k-1}, v^{t,k-1}, \lambda^{t,k-1}) + D_\lambda^{t,k-1}$$

Outer loop index

Inner loop index

Unbiased estimators of the directions

SRBA (Stochastic Recursive Bilevel Algorithm)

General principle

- Adaptation of SARAH/SPIDER to bilevel setting [Nguyen et al. '17, Fang et al. '18]
- Recursive estimate of the directions
- Periodic reinitialization of the estimate

Reinitialization of estimate directions

$$D_\theta^{t,0} = D_\theta(\theta^{t,0}, v^{t,0}, \lambda^{t,0})$$
$$D_v^{t,0} = D_v(\theta^{t,0}, v^{t,0}, \lambda^{t,0})$$
$$D_\lambda^{t,0} = D_\lambda(\theta^{t,}, v^{t,0}, \lambda^{t,0})$$

Full batch directions

Recursive estimation of the directions

Sample $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$ and set

$$D_\theta^{t,k} = D_\theta^{i,j}(\theta^{t,k}, v^{t,k}, \lambda^{t,k}) - D_\theta^{i,j}(\theta^{t,k-1}, v^{t,k-1}, \lambda^{t,k-1}) + D_\theta^{t,k-1}$$
$$D_v^{t,k} = D_v^{i,j}(\theta^{t,k}, v^{t,k}, \lambda^{t,k}) - D_v^{i,j}(\theta^{t,k-1}, v^{t,k-1}, \lambda^{t,k-1}) + D_v^{t,k-1}$$
$$D_\lambda^{t,k} = D_\lambda^{i,j}(\theta^{t,k}, v^{t,k}, \lambda^{t,k}) - D_\lambda^{i,j}(\theta^{t,k-1}, v^{t,k-1}, \lambda^{t,k-1}) + D_\lambda^{t,k-1}$$

Outer loop index

Inner loop index

Unbiased estimators of the directions

Convergence of SRBA

Convergence of SRBA

Theorem

Assume that

Convergence of SRBA

Theorem

Assume that

1. the outer function f is twice differentiable with Lipschitz derivatives

Convergence of SRBA

Theorem

Assume that

1. the outer function f is twice differentiable with Lipschitz derivatives
2. the inner function g is three times differentiable with Lipschitz derivatives

Convergence of SRBA

Theorem

Assume that

1. the outer function f is twice differentiable with Lipschitz derivatives
2. the inner function g is three times differentiable with Lipschitz derivatives

Then, for constant step sizes proportional to $(n+m)^{-\frac{1}{2}}$,

$$\mathcal{O}((n+m)^{\frac{1}{2}}\epsilon^{-1})$$

calls to oracles are sufficient to find an ϵ-stationary point

Convergence of SRBA

Theorem

Assume that

1. the outer function f is twice differentiable with Lipschitz derivatives
2. the inner function g is three times differentiable with Lipschitz derivatives

Then, for constant step sizes proportional to $(n+m)^{-\frac{1}{2}}$,

$$\mathcal{O}((n+m)^{\frac{1}{2}}\epsilon^{-1})$$

calls to oracles are sufficient to find an ϵ-stationary point

Similar to the rate of SARAH for non-convex smooth finite sums [Nguyen '22]

Lower bound for bilevel empirical risk minimization

M. Dagréou, T. Moreau, S. Vaiter, P. Ablin. A Lower Bound and a Near-Optimal Algorithm for Bilevel Empirical Risk Minimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024.

Complexity bounds in optimization

Question

What is the amount of oracle computations I need to solve bilevel ERM with smooth outer and strongly convex inner functions by only accessing individual gradient of the outer function and gradient/HVP/JVP of the inner function?

Complexity bounds in optimization

Proxy of the total amount of elementary operations of an algorithm

Question

What is the amount of oracle computations I need to solve bilevel ERM with smooth outer and strongly convex inner functions by only accessing individual gradient of the outer function and gradient/HVP/JVP of the inner function?

Complexity bounds in optimization

Proxy of the total amount of elementary operations of an algorithm

Class of problems

Question

What is the amount of oracle computations I need to solve bilevel ERM with smooth outer and strongly convex inner functions by only accessing individual gradient of the outer function and gradient/HVP/JVP of the inner function?

Complexity bounds in optimization

Proxy of the total amount of elementary operations of an algorithm

Class of problems

Question

What is the amount of oracle computations I need to solve bilevel ERM with smooth outer and strongly convex inner functions by only accessing individual gradient of the outer function and gradient/HVP/JVP of the inner function?

Class of algorithms

A detour by single-level optimization

A detour by single-level optimization

Finite sum minimization setting

$$\min_{x\in\mathbb{R}^d} h(x) = \frac{1}{n}\sum_{i=1}^{n} h_i(x)$$

A detour by single-level optimization

Finite sum minimization setting

$$\min_{x\in\mathbb{R}^d} h(x) = \frac{1}{n}\sum_{i=1}^{n} h_i(x)$$

L-smooth

A detour by single-level optimization

Finite sum minimization setting

$$\min_{x\in\mathbb{R}^d} h(x) = \frac{1}{n}\sum_{i=1}^{n} h_i(x)$$

L-smooth

Algorithm class

$$x^{t+1} \in x^0 + \mathrm{span}\left\{\nabla h_{i_0}(x^0), \cdots, \nabla h_{i_t}(x^t)\right\}$$

A detour by single-level optimization

Finite sum minimization setting

$$\min_{x \in \mathbb{R}^d} h(x) = \frac{1}{n} \sum_{i=1}^{n} h_i(x)$$

L-smooth

Algorithm class

$$x^{t+1} \in x^0 + \operatorname{span}\left\{\nabla h_{i_0}(x^0), \cdots, \nabla h_{i_t}(x^t)\right\}$$

Random variables in $\{1, \ldots, n\}$

A detour by single-level optimization

Finite sum minimization setting

$$\min_{x \in \mathbb{R}^d} h(x) = \frac{1}{n} \sum_{i=1}^{n} h_i(x)$$

L-smooth

Algorithm class

$$x^{t+1} \in x^0 + \operatorname{span}\left\{\nabla h_{i_0}(x^0), \cdots, \nabla h_{i_t}(x^t)\right\}$$

Random variables in $\{1, \ldots, n\}$

Upper bound [Nguyen '22]
There exists an algorithm which is able to find an ϵ-stationary point of any function h in less than

$$\mathcal{O}(\sqrt{n}\epsilon^{-1})$$

oracle calls.

A detour by single-level optimization

Finite sum minimization setting

$$\min_{x \in \mathbb{R}^d} h(x) = \frac{1}{n} \sum_{i=1}^{n} h_i(x)$$

L-smooth

Upper bound [Nguyen '22]

There exists an algorithm which is able to find an ϵ-stationary point of any function h in less than

$$\mathcal{O}(\sqrt{n}\epsilon^{-1})$$

oracle calls.

Algorithm class

$$x^{t+1} \in x^0 + \operatorname{span}\left\{\nabla h_{i_0}(x^0), \cdots, \nabla h_{i_t}(x^t)\right\}$$

Random variables in $\{1, \ldots, n\}$

Lower bound [Zhou et al. '19]

Given an algorithm A we can find a function h such that A requires at least

$$\Omega(\sqrt{n}\epsilon^{-1})$$

oracle calls to find an ϵ-stationary point.

Why does single-level results do not extend directly to bilevel problems?

Why does single-level results do not extend directly to bilevel problems?

A single-level problem is a bilevel problem

$$\min_{x \in \mathbb{R}^d} h(x)$$

Why does single-level results do not extend directly to bilevel problems?

A single-level problem is a bilevel problem

$$\min_{x\in\mathbb{R}^d} \Phi(x) = h(y^*(x))$$

$$y^*(x) \in \operatorname*{argmin}_{y\in\mathbb{R}^d} \frac{1}{2}\|y-x\|^2$$

Why does single-level results do not extend directly to bilevel problems?

A single-level problem is a bilevel problem

$$\min_{x \in \mathbb{R}^d} \Phi(x) = h(y^*(x))$$

$$y^*(x) \in \operatorname*{argmin}_{y \in \mathbb{R}^d} \frac{1}{2}\|y - x\|^2$$

We could expect a higher lower bound

Why does single-level results do not extend directly to bilevel problems?

A single-level problem is a bilevel problem

$$\min_{x\in\mathbb{R}^d} \Phi(x) = h(y^*(x))$$

$$y^*(x) \in \operatorname*{argmin}_{y\in\mathbb{R}^d} \frac{1}{2}\|y-x\|^2$$

Algorithm classes

We could expect a higher lower bound

Why does single-level results do not extend directly to bilevel problems?

A single-level problem is a bilevel problem

$$\min_{x\in\mathbb{R}^d} \Phi(x) = h(y^*(x))$$

$$y^*(x) \in \operatorname*{argmin}_{y\in\mathbb{R}^d} \frac{1}{2}\|y - x\|^2$$

Algorithm classes

- Single-level analysis assumes that we sample gradients of Φ

We could expect a higher lower bound

Why does single-level results do not extend directly to bilevel problems?

A single-level problem is a bilevel problem

$$\min_{x\in\mathbb{R}^d} \Phi(x) = h(y^*(x))$$

$$y^*(x) \in \operatorname*{argmin}_{y\in\mathbb{R}^d} \frac{1}{2}\|y-x\|^2$$

Algorithm classes

- Single-level analysis assumes that we sample gradients of Φ
- Classical bilevel algorithms do not have access to the exact value of this gradient

We could expect a higher lower bound

Why does single-level results do not extend directly to bilevel problems?

A single-level problem is a bilevel problem

$$\min_{x \in \mathbb{R}^d} \Phi(x) = h(y^*(x))$$

$$y^*(x) \in \operatorname*{argmin}_{y \in \mathbb{R}^d} \frac{1}{2}\|y - x\|^2$$

We could expect a higher lower bound

Algorithm classes

- Single-level analysis assumes that we sample gradients of Φ
- Classical bilevel algorithms do not have access to the exact value of this gradient

We need a specific algorithm class

Algorithm class

Update directions

- $D_\theta(\theta, v, \lambda) = \nabla_\theta g(\lambda, \theta)$
- $D_v(\theta, v, \lambda) = \nabla^2_{\theta,\theta} g(\lambda, \theta) v + \nabla_\theta f(\lambda, \theta)$
- $D_\lambda(\theta, v, \lambda) = \nabla^2_{\lambda,\theta} g(\lambda, \theta) v + \nabla_\lambda f(\lambda, \theta)$

Algorithm class

Update directions

- $D_\theta(\theta, v, \lambda) = \nabla_\theta g(\lambda, \theta)$
- $D_v(\theta, v, \lambda) = \nabla^2_{\theta,\theta} g(\lambda, \theta) v + \nabla_\theta f(\lambda, \theta)$
- $D_\lambda(\theta, v, \lambda) = \nabla^2_{\lambda,\theta} g(\lambda, \theta) v + \nabla_\lambda f(\lambda, \theta)$

Linear Bilevel Algorithm

$$\theta^{t+1} \in \theta^0 + \operatorname{span}\left\{\nabla_\theta g_{i_0}(\lambda^0, \theta^0), \ldots, \nabla_\theta g_{i_t}(\lambda^t, \theta^t)\right\}$$

$$\begin{aligned} v^{t+1} \in v^0 + \operatorname{span}\{&\nabla^2_{\theta,\theta} g_{i_0}(\lambda^0, \theta^0) v^0 + \nabla_\theta f_{j_0}(\lambda^0, \theta^0), \\ &\ldots, \nabla_\theta g_{i_t}(\lambda^t, \theta^t) v^t + \nabla_\theta f_{j_t}(\lambda^t, \theta^t)\} \end{aligned}$$

$$\begin{aligned} \lambda^{t+1} \in \lambda^0 + \operatorname{span}\{&\nabla^2_{\lambda,\theta} g_{i_0}(\lambda^0, \theta^0) v^0 + \nabla_\lambda f_{j_0}(\lambda^0, \theta^0), \\ &\ldots, \nabla^2_{\lambda,\theta} g_{i_t}(\lambda^t, \theta^t) v^t + \nabla_\lambda f_{j_t}(\lambda^t, \theta^t)\} \end{aligned}$$

Algorithm class

Update directions

- $D_\theta(\theta, v, \lambda) = \nabla_\theta g(\lambda, \theta)$
- $D_v(\theta, v, \lambda) = \nabla^2_{\theta,\theta} g(\lambda, \theta) v + \nabla_\theta f(\lambda, \theta)$
- $D_\lambda(\theta, v, \lambda) = \nabla^2_{\lambda,\theta} g(\lambda, \theta) v + \nabla_\lambda f(\lambda, \theta)$

- Contains several several bilevel algorithms

Linear Bilevel Algorithm

$$\theta^{t+1} \in \theta^0 + \operatorname{span}\left\{\nabla_\theta g_{i_0}(\lambda^0, \theta^0), \ldots, \nabla_\theta g_{i_t}(\lambda^t, \theta^t)\right\}$$

$$\begin{aligned} v^{t+1} \in v^0 + \operatorname{span}\{&\nabla^2_{\theta,\theta} g_{i_0}(\lambda^0, \theta^0) v^0 + \nabla_\theta f_{j_0}(\lambda^0, \theta^0), \\ &\ldots, \nabla_\theta g_{i_t}(\lambda^t, \theta^t) v^t + \nabla_\theta f_{j_t}(\lambda^t, \theta^t)\} \end{aligned}$$

$$\begin{aligned} \lambda^{t+1} \in \lambda^0 + \operatorname{span}\{&\nabla^2_{\lambda,\theta} g_{i_0}(\lambda^0, \theta^0) v^0 + \nabla_\lambda f_{j_0}(\lambda^0, \theta^0), \\ &\ldots, \nabla^2_{\lambda,\theta} g_{i_t}(\lambda^t, \theta^t) v^t + \nabla_\lambda f_{j_t}(\lambda^t, \theta^t)\} \end{aligned}$$

Algorithm class

Update directions

- $D_\theta(\theta, v, \lambda) = \nabla_\theta g(\lambda, \theta)$
- $D_v(\theta, v, \lambda) = \nabla^2_{\theta,\theta} g(\lambda, \theta) v + \nabla_\theta f(\lambda, \theta)$
- $D_\lambda(\theta, v, \lambda) = \nabla^2_{\lambda,\theta} g(\lambda, \theta) v + \nabla_\lambda f(\lambda, \theta)$

- Contains several several bilevel algorithms
- But excludes non-linear subroutines like Neumann iterations

Linear Bilevel Algorithm

$$\theta^{t+1} \in \theta^0 + \mathrm{span}\left\{\nabla_\theta g_{i_0}(\lambda^0, \theta^0), \ldots, \nabla_\theta g_{i_t}(\lambda^t, \theta^t)\right\}$$

$$v^{t+1} \in v^0 + \mathrm{span}\left\{\nabla^2_{\theta,\theta} g_{i_0}(\lambda^0, \theta^0) v^0 + \nabla_\theta f_{j_0}(\lambda^0, \theta^0), \ldots, \nabla_\theta g_{i_t}(\lambda^t, \theta^t) v^t + \nabla_\theta f_{j_t}(\lambda^t, \theta^t)\right\}$$

$$\lambda^{t+1} \in \lambda^0 + \mathrm{span}\left\{\nabla^2_{\lambda,\theta} g_{i_0}(\lambda^0, \theta^0) v^0 + \nabla_\lambda f_{j_0}(\lambda^0, \theta^0), \ldots, \nabla^2_{\lambda,\theta} g_{i_t}(\lambda^t, \theta^t) v^t + \nabla_\lambda f_{j_t}(\lambda^t, \theta^t)\right\}$$

Problem class

Problem class

Bilevel Optimization Problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) \in \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Problem class

Bilevel Optimization Problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) \in \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Empirical Risk Minimization

$$f(\lambda, \theta) = \frac{1}{m} \sum_{j=1}^{m} f_j(\lambda, \theta), \quad g(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} g_i(\lambda, \theta)$$

Problem class

Bilevel Optimization Problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) \in \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Empirical Risk Minimization

$$f(\lambda, \theta) = \frac{1}{m} \sum_{j=1}^{m} f_j(\lambda, \theta), \quad g(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} g_i(\lambda, \theta)$$

Lipschitz gradient

Problem class

Bilevel Optimization Problem

$$\min_{\lambda \in \mathbb{R}^{d_\lambda}} \Phi(\lambda) \triangleq f(\lambda, \theta^*(\lambda))$$

$$\theta^*(\lambda) \in \operatorname*{argmin}_{\theta \in \mathbb{R}^{d_\theta}} g(\lambda, \theta)$$

Empirical Risk Minimization

$$f(\lambda, \theta) = \frac{1}{m} \sum_{j=1}^{m} f_j(\lambda, \theta), \quad g(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} g_i(\lambda, \theta)$$

Lipschitz gradient

Twice differentiable,
Strongly convex

Lower bound for bilevel ERM

Theorem (informal)

For any linear bilevel algorithm, for a large enough dimension d_λ we can find an instantiation of bilevel ERM problem such that such that finding a point $\hat{\lambda} \in \mathbb{R}^{d_\lambda}$ that verifies

$$\mathbb{E}[\|\nabla\Phi(\hat{\lambda})\|^2] \leq \epsilon$$

requires at least $\Omega(m^{\frac{1}{2}}\epsilon^{-1})$ gradient/HVP/JVP computations.

Lower bound for bilevel ERM

Theorem (informal)

For any linear bilevel algorithm, for a large enough dimension d_λ we can find an instantiation of bilevel ERM problem such that such that finding a point $\hat{\lambda} \in \mathbb{R}^{d_\lambda}$ that verifies

$$\mathbb{E}[\|\nabla\Phi(\hat{\lambda})\|^2] \leq \epsilon$$

requires at least $\Omega(m^{\frac{1}{2}}\epsilon^{-1})$ gradient/HVP/JVP computations.

- Similar to result of finite sum minimization in nonconvex setting [Zhou et al. '19]

Lower bound for bilevel ERM

Theorem (informal)

For any linear bilevel algorithm, for a large enough dimension d_λ we can find an instantiation of bilevel ERM problem such that such that finding a point $\hat{\lambda} \in \mathbb{R}^{d_\lambda}$ that verifies

$$\mathbb{E}[\|\nabla\Phi(\hat{\lambda})\|^2] \leq \epsilon$$

requires at least $\Omega(m^{\frac{1}{2}}\epsilon^{-1})$ gradient/HVP/JVP computations.

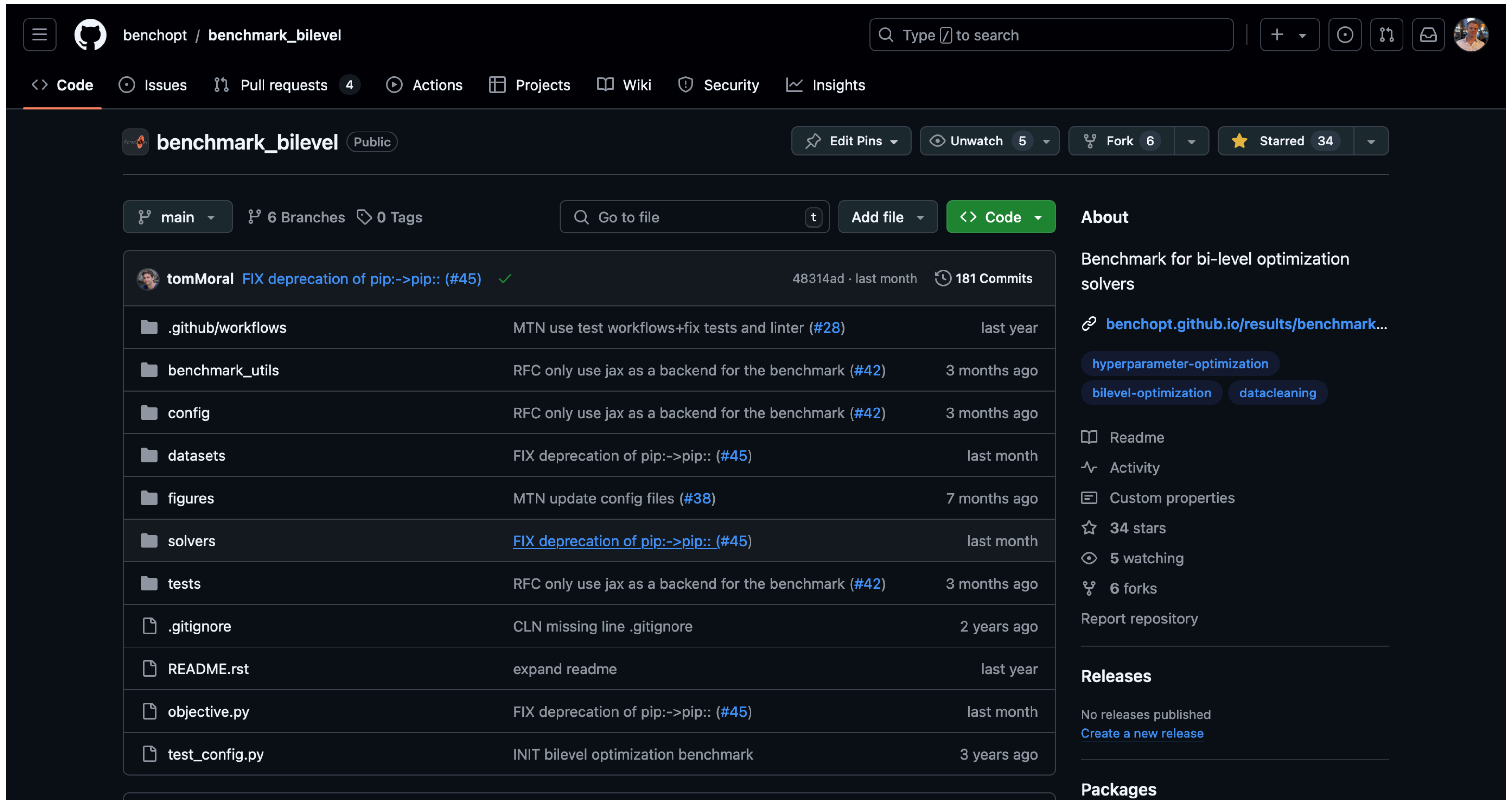
- Similar to result of finite sum minimization in nonconvex setting [Zhou et al. '19]
- Still missing the dependency on the inner number of samples

Numerical evaluation of bilevel algorithms

Benchmark of bilevel algorithms

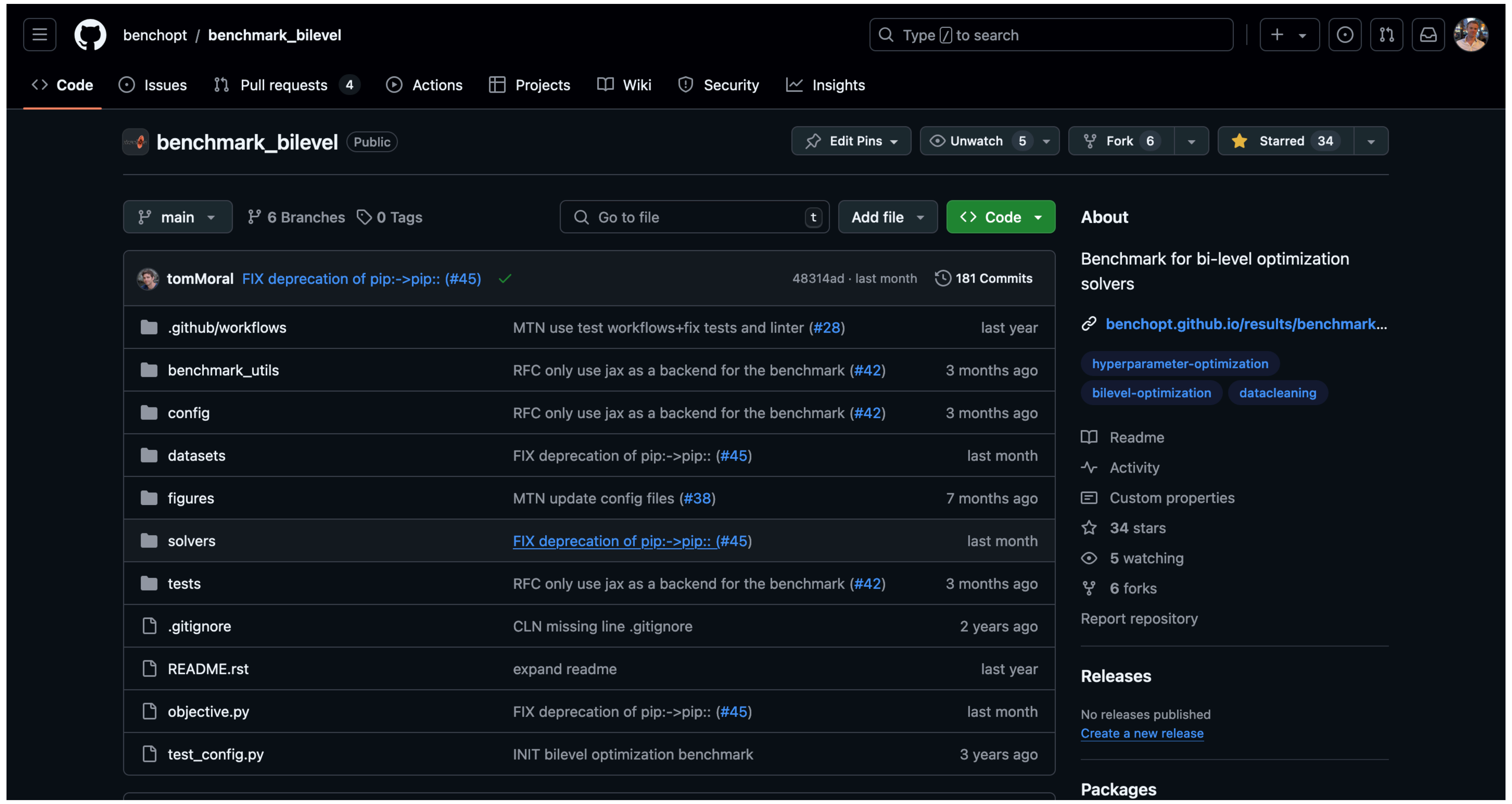
Benchmark of bilevel algorithms

- Open and reproducible benchmark



Benchmark of bilevel algorithms

- Open and reproducible benchmark
- Benchopt ecosystem, Jax framework

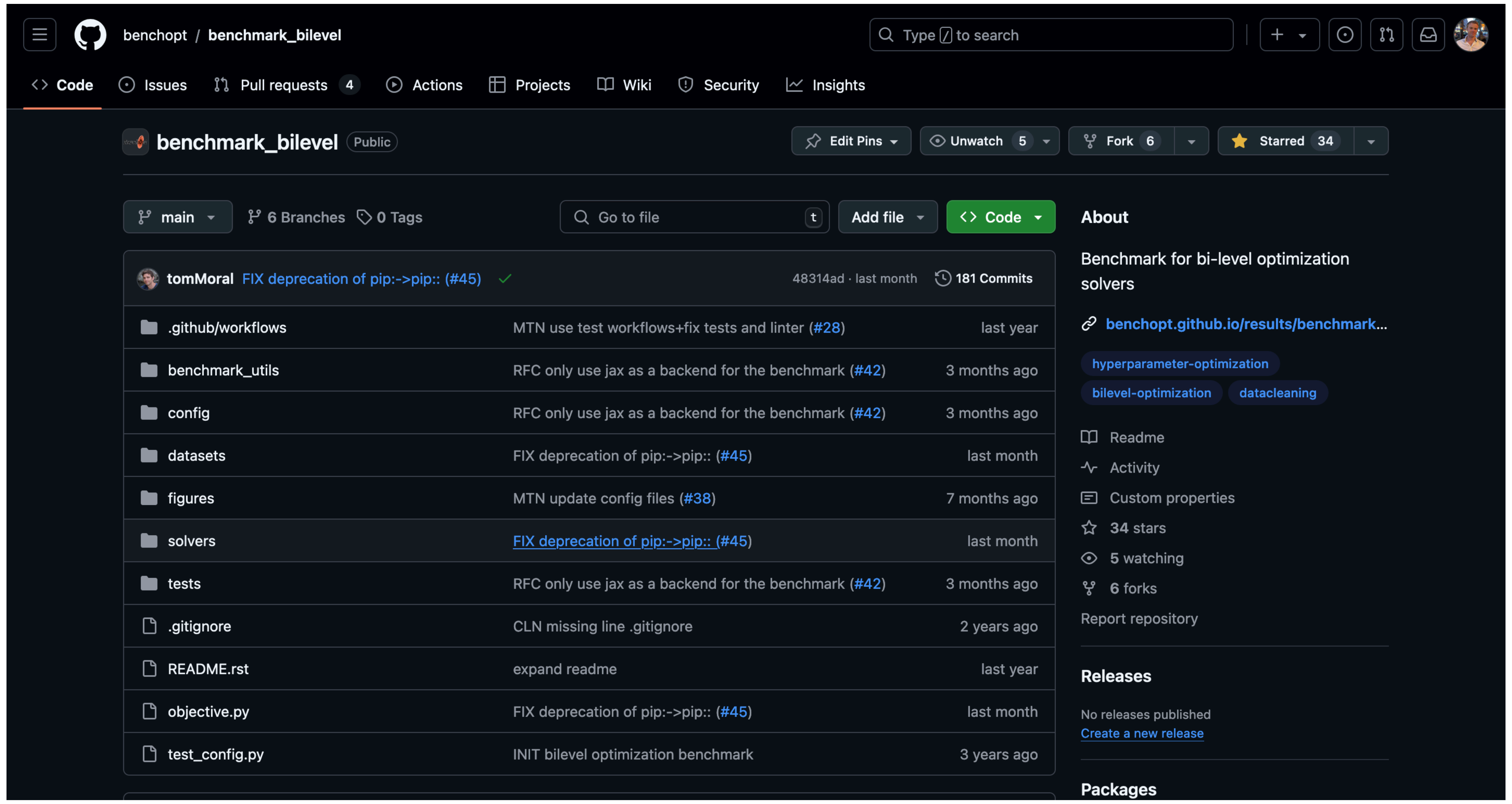


T. Moreau et al. *Benchopt: Reproducible, efficient and collaborative optimization benchmarks*. In Advances in Neural Information Processing Systems (NeurIPS), 2022.

Benchmark of bilevel algorithms

Benchopt

- Open and reproducible benchmark
- Benchopt ecosystem, Jax framework
- 17 solvers: stochastic, deterministic, variance reduction, Hessian free…

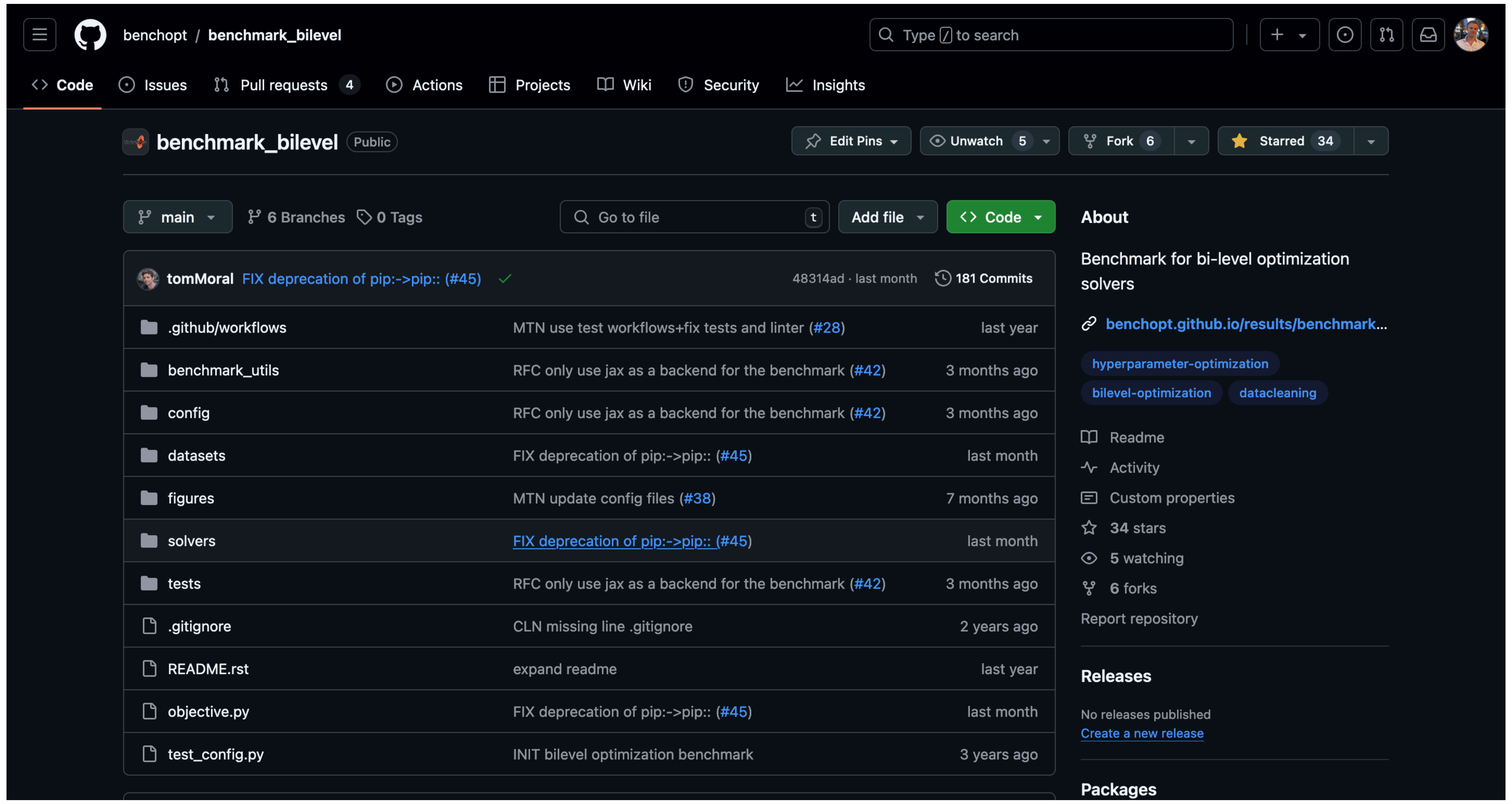


T. Moreau et al. *Benchopt: Reproducible, efficient and collaborative optimization benchmarks*. In Advances in Neural Information Processing Systems (NeurIPS), 2022.

Benchmark of bilevel algorithms

Benchopt

- Open and reproducible benchmark
- Benchopt ecosystem, Jax framework
- 17 solvers: stochastic, deterministic, variance reduction, Hessian free…
- 4 tasks: quadratics, hyperparameter selection with ICJNN1 and COVTYPE, data hypercleaning with MNIST



T. Moreau et al. *Benchopt: Reproducible, efficient and collaborative optimization benchmarks*. In Advances in Neural Information Processing Systems (NeurIPS), 2022.

Data hypercleaning [Franceschi et al. '17]

Setting

Data hypercleaning [Franceschi et al. ’17]

Setting

- Dataset: MNIST

Data hypercleaning [Franceschi et al. ’17]

Setting

- Dataset: MNIST
- Training samples with corrupted labels

Data hypercleaning [Franceschi et al. '17]

Setting

- Dataset: MNIST
- Training samples with corrupted labels
- Idea: Give more weight to uncorrupted samples

$$g(\lambda, \theta) = \frac{1}{n}\sum_{i=1}^{n} \sigma(\lambda_i)\ell(\theta x_i^{\text{train}}, y_i^{\text{train}}) + C_r \|\theta\|^2$$

with $\sigma(\lambda_i) \in [0, 1]$

Data hypercleaning [Franceschi et al. '17]

Setting

- Dataset: MNIST
- Training samples with corrupted labels
- Idea: Give more weight to uncorrupted samples

$$g(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} \sigma(\lambda_i) \ell(\theta x_i^{\mathrm{train}}, y_i^{\mathrm{train}}) + C_r \|\theta\|^2$$

with $\sigma(\lambda_i) \in [0, 1]$

- Weights are tuned by minimizing validation loss

Data hypercleaning [Franceschi et al. '17]

Setting

- Dataset: MNIST

- Training samples with corrupted labels

- Idea: Give more weight to uncorrupted samples

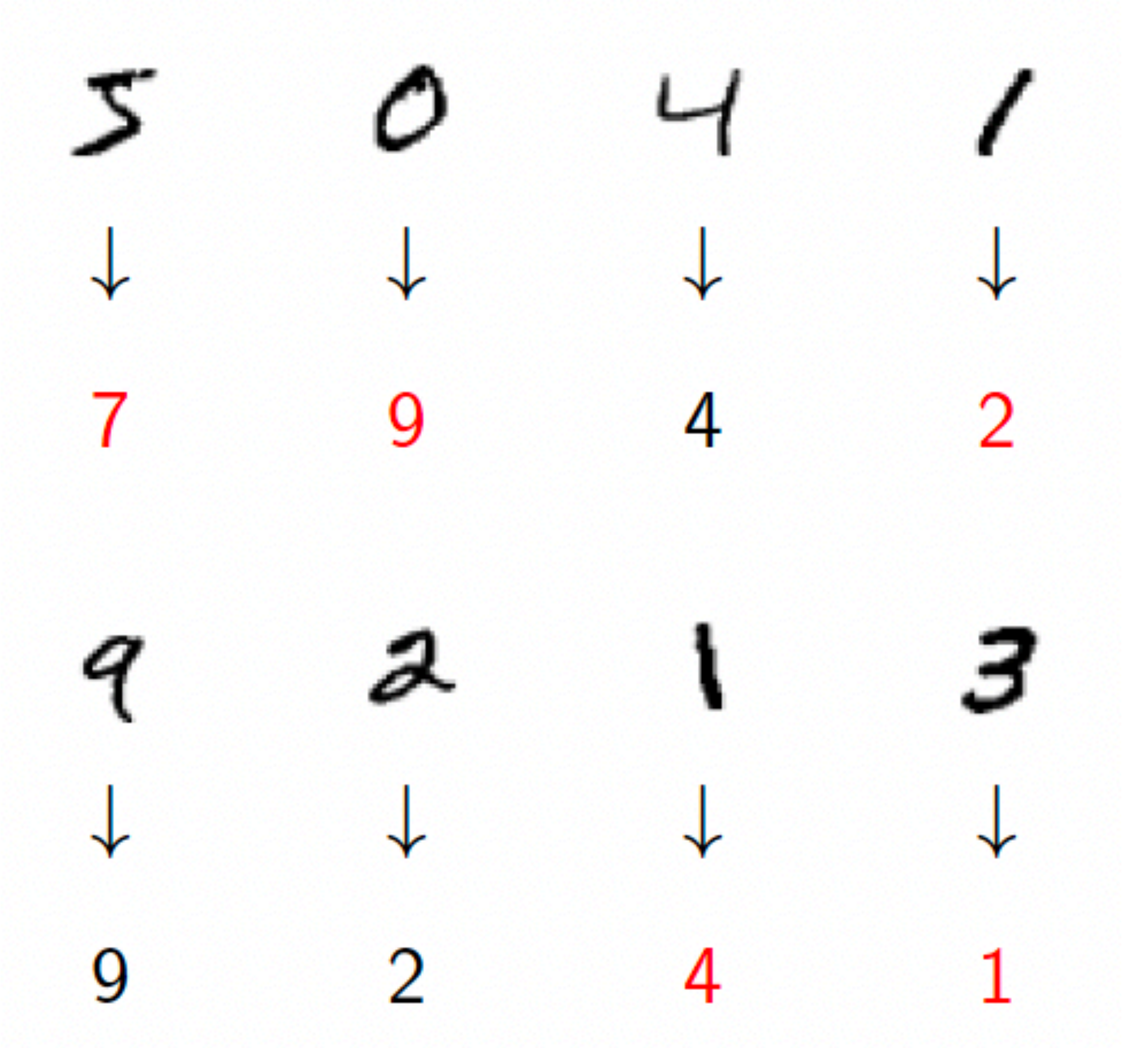
$$g(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} \sigma(\lambda_i) \ell(\theta x_i^{\text{train}}, y_i^{\text{train}}) + C_r \|\theta\|^2$$

with $\sigma(\lambda_i) \in [0, 1]$

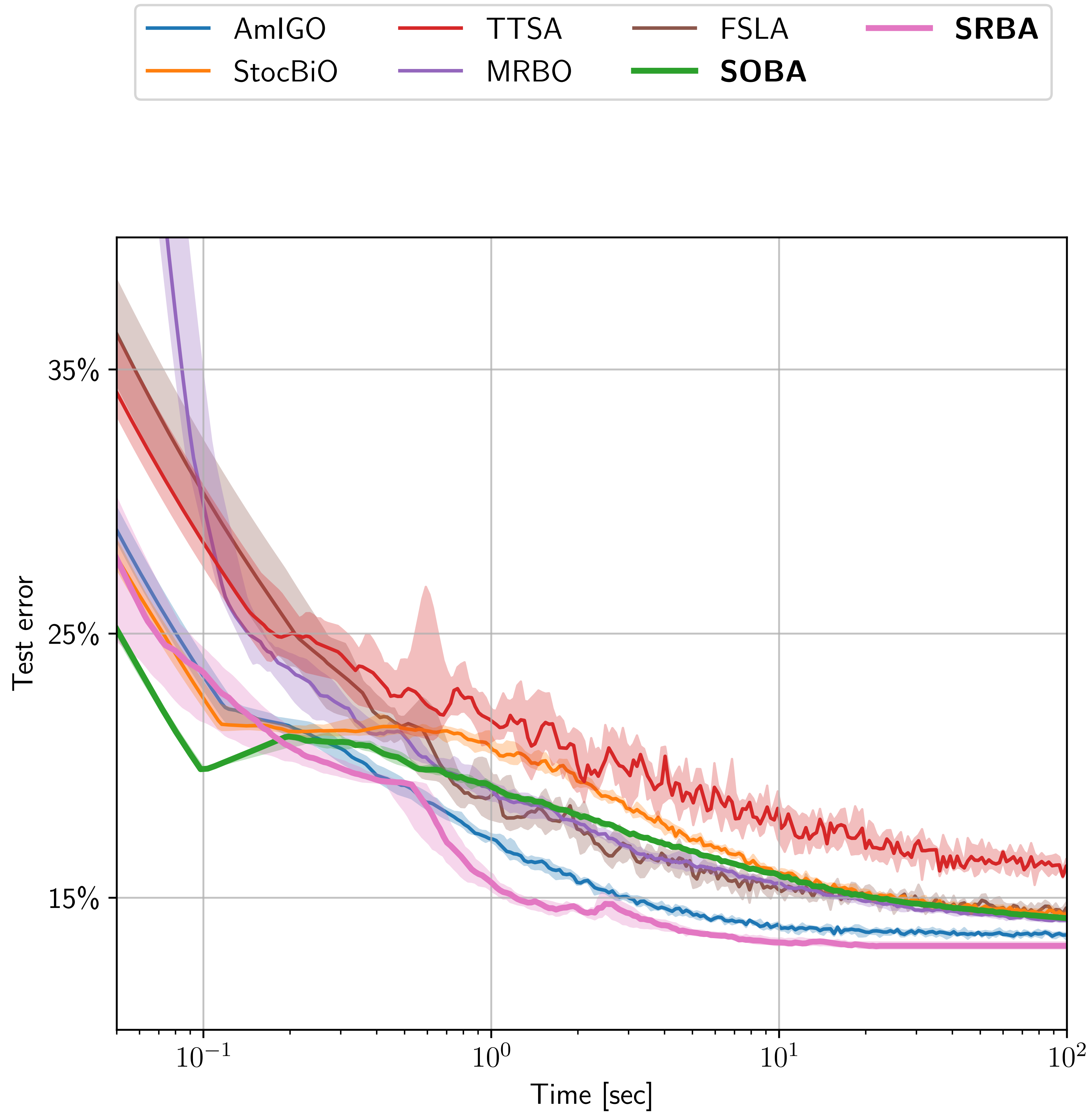
- Weights are tuned by minimizing validation loss

$$f(\theta^*(\lambda)) = \frac{1}{m} \sum_{j=1}^{m} \ell(\theta^*(\lambda) x_i^{\text{val}}, y_i^{\text{val}})$$

Correct labels



Data hypercleaning



Conclusion and perpectives

Conclusion

Conclusion

- Provided a modular algorithmic framework that enables to adapt single-level techniques to the bilevel setting

Conclusion

- Provided a modular algorithmic framework that enables to adapt single-level techniques to the bilevel setting
- Instantiations of this framework yields similar oracle complexity to their single-level counterparts.

Conclusion

- Provided a modular algorithmic framework that enables to adapt single-level techniques to the bilevel setting
- Instantiations of this framework yields similar oracle complexity to their single-level counterparts.
- Provided a complexity lower bound for bilevel problems.

Conclusion

- Provided a modular algorithmic framework that enables to adapt single-level techniques to the bilevel setting
- Instantiations of this framework yields similar oracle complexity to their single-level counterparts.
- Provided a complexity lower bound for bilevel problems.
- Provided an open benchmark to compare bilevel algorithms

Perspectives/Open questions

Perspectives/Open questions

- Sensitivity to the step sizes choice

Perspectives/Open questions

- Sensitivity to the step sizes choice

- Generalization performances of gradient-based hyperparameter selection procedures

Perspectives/Open questions

- Sensitivity to the step sizes choice
- Generalization performances of gradient-based hyperparameter selection procedures
- Understanding performances of implicit differentiation-based techniques when apply to problems with non-strongly convex inner functions

Thanks for your attention!!!

Conference papers

- **M. Dagréou**, P. Ablin, S. Vaiter, T. Moreau. A framework for bilevel optimization that enables stochastic and global variance reduction algorithm. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. *Oral*

- **M. Dagréou**, T. Moreau, S. Vaiter, P. Ablin. A Lower Bound and a Near-Optimal Algorithm for Bilevel Empirical Risk Minimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024.

- T. Moreau et al. Benchopt: Reproducible, efficient and collaborative optimization benchmarks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Miscellaneous

- **M. Dagréou**, P. Ablin, S. Vaiter, T. Moreau. How to compute Hessian-vector products? In *ICLR blogpost track*, 2024. *Spotlight*