Bilevel Optimization: Convergence Analysis and Enhanced Design

Kaiyi Ji, Junjie Yang, Yingbin Liang

Electrical Computer Engineering

Ohio State University

06/20/2021

Bilevel Optimization in ML

- Few-shot meta-learning

Image sources: Ravi & Larochelle, 2017

Bilevel procedure:

- ➢ Inner level: **each task** adapts own parameters
- ➢ Outer level: update **embedding model** for **all tasks**

Bilevel Optimization in ML

- Hyperparameter Optimization:

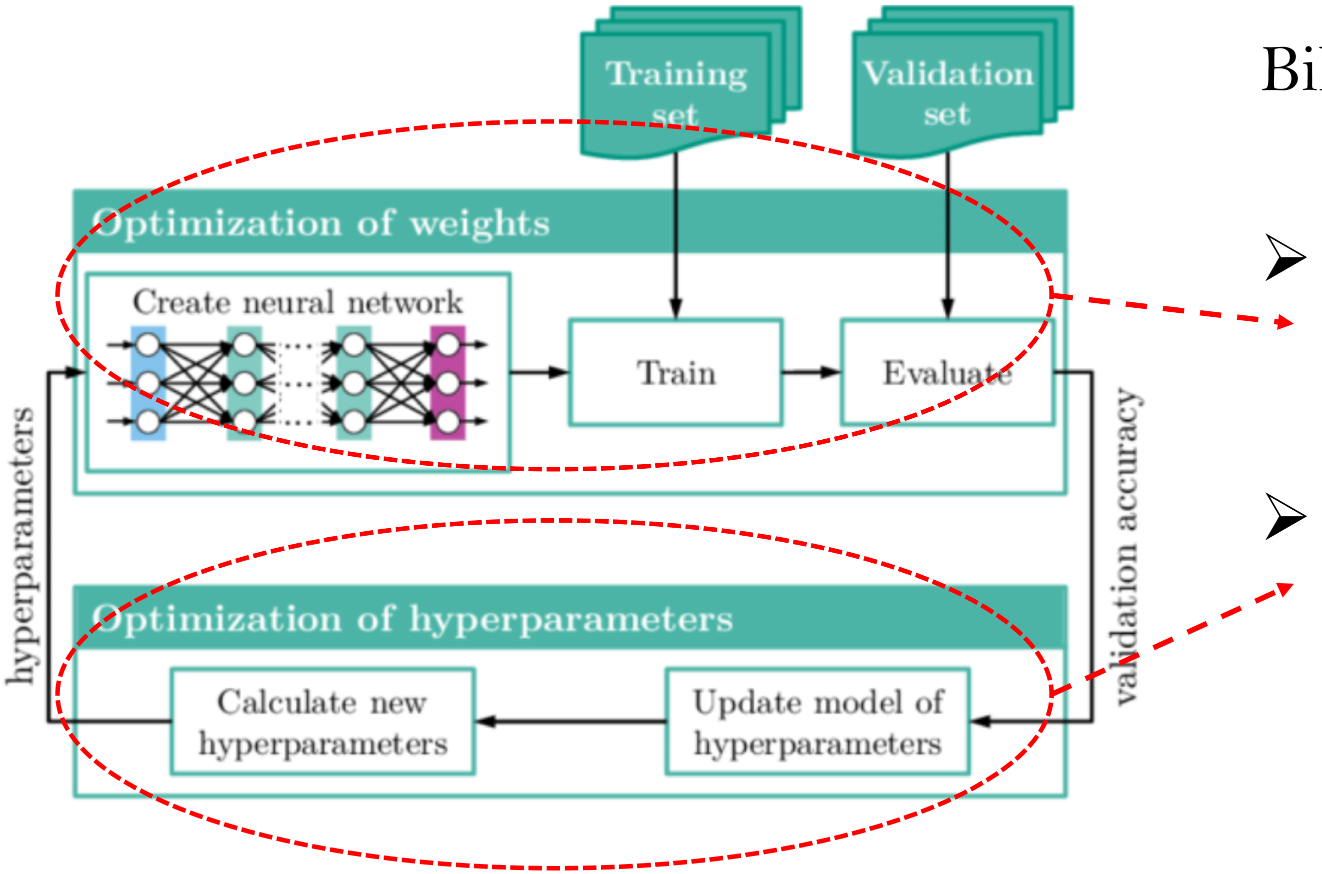


Image sources: Stock, 2020

Bilevel procedure:

- Inner level: optimize model **weights** on **training data**
- Outer level: update (finetune) **hyperparameters** on **validation data**

Problem Formulation

- Objective function

$$\min_{x\in\mathbb{R}^p} \Phi(x) := f(x, y^*(x))$$
$$\text{s.t.} \quad y^*(x) = \arg\min_{y\in\mathbb{R}^q} g(x, y),$$

> - $f(x, y)$: outer-level loss; $g(x, y)$: inner-level loss
> - $y^*(x)$: minimizer of inner-level loss $g(x, \cdot)$

- Applications:
  - Meta-learning:
    - ❑ x: embedding model parameters
    - ❑ y: task-specific weights
  - Hyperparameter optimization:
    - ❑ x: hyperparameters
    - ❑ y: model weights

Hypergradient & Existing Methods

- Hypergradient: $\nabla\Phi(x) = \frac{\partial f(x, y^*(x))}{\partial x}$
- Two major classes
  - Approximate **implicit differentiation** (AID):

$$\nabla\Phi(x) = \nabla_x f(x, y^*(x)) - \nabla_x \nabla_y g(x, y^*(x))[\nabla_y^2 g(x, y^*(x))]^{-1} \nabla_y f(x, y^*(x)).$$

    - Approximate Hessian-inverse-vector via solving linear system

  - Iterative differentiation (ITD):
    - Compute $y^N(x)$ via N steps of iterative algorithms

$$\frac{\partial f(x, y^N(x))}{\partial x} \to \nabla\Phi(x) = \frac{\partial f(x, y^*(x))}{\partial x}$$

Open Questions

- Limited **non-asymptotic** analysis
  - Whether they converge in finite steps for most applications
  - No quantitative comparison among these algorithms
  - No guidelines for parameter selection

- AID-based methods:
  - Existing analysis: **increasing number** of inner-level steps

    Practice: constant number | Theory: worse rate

- ITD-based methods:
  - No convergence rate analysis yet

Open Questions

- Loss functions often take a finite-sum form

$$f(x,y) = \frac{1}{n}\sum_{i=1}^{n} F(x,y;\xi_i)$$

$$g(x,y) = \frac{1}{m}\sum_{i=1}^{m} G(x,y;\zeta_i)$$

  - ξ_i, ζ_i: data samples

How to design a principled algorithm in sampling setting?

Can **stochastic data sampling** improve efficiency?

Our Results:

- Theory

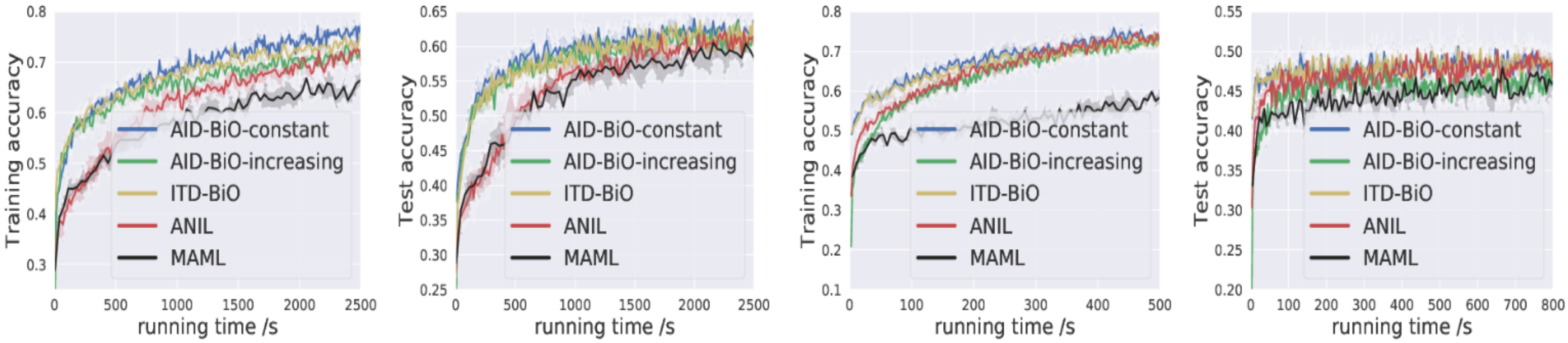
Algorithm	$\mathrm{Gc}(f,\epsilon)$	$\mathrm{Gc}(g,\epsilon)$	$\mathrm{JV}(g,\epsilon)$	$\mathrm{HV}(g,\epsilon)$
AID-BiO (Ghadimi & Wang, 2018)	$\mathcal{O}(\kappa^4\epsilon^{-1})$	$\mathcal{O}(\kappa^5\epsilon^{-5/4})$	$\mathcal{O}\left(\kappa^4\epsilon^{-1}\right)$	$\widetilde{\mathcal{O}}\left(\kappa^{4.5}\epsilon^{-1}\right)$
AID-BiO (this paper)	$\mathcal{O}(\kappa^3\epsilon^{-1})$	$\mathcal{O}(\kappa^4\epsilon^{-1})$	$\mathcal{O}\left(\kappa^3\epsilon^{-1}\right)$	$\mathcal{O}\left(\kappa^{3.5}\epsilon^{-1}\right)$
ITD-BiO (this paper)	$\mathcal{O}(\kappa^3\epsilon^{-1})$	$\widetilde{\mathcal{O}}(\kappa^4\epsilon^{-1})$	$\widetilde{\mathcal{O}}\left(\kappa^4\epsilon^{-1}\right)$	$\widetilde{\mathcal{O}}\left(\kappa^4\epsilon^{-1}\right)$

$\mathrm{Gc}(f,\epsilon)$ and $\mathrm{Gc}(g,\epsilon)$: number of gradient evaluations w.r.t. f and g.
$\mathrm{JV}(g,\epsilon)$: number of Jacobian-vector products.
$\mathrm{HV}(g,\epsilon)$: number of Hessian-vector products.

- Improved complexity over AID-BiO
  - Constant inner-level steps
  - Inner-level warm start

- First result on ITD-BiO

Experiments on Meta-Learning



(a) dataset: miniImageNet

(b) dataset: FC100

- ➢ Our AID-BiO-constant performs best

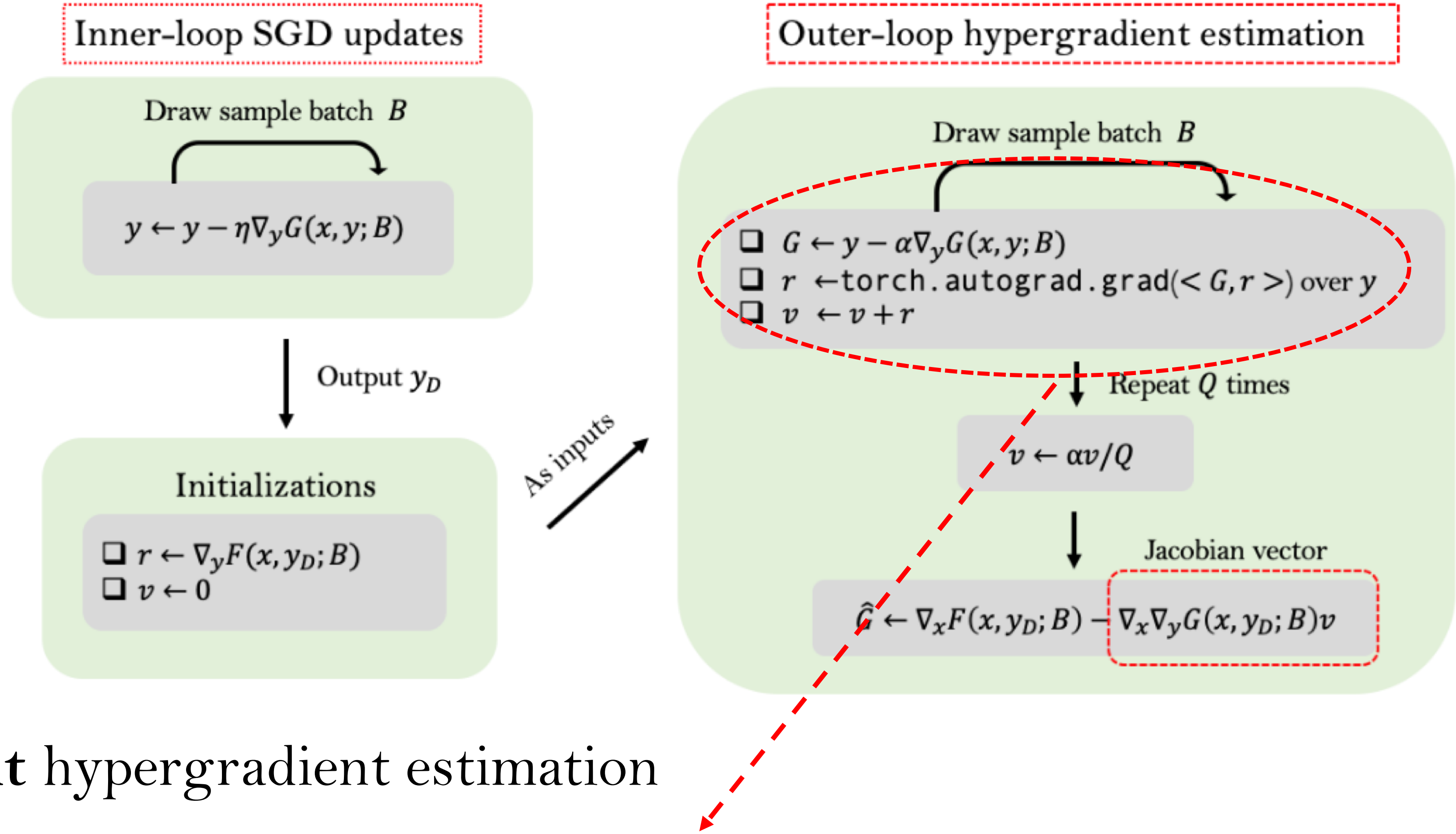
- ➢ Our repo: https://github.com/JunjieYang97/stocBiO

Check!

❑ More efficient **first-order** ITD-BiO is developed!

Fast Stochastic Bilevel Optimizer

- **Stochastic** bilevel optimizer: StocBiO



- **Efficient** hypergradient estimation
  - ➢ only involve **Hessian- and Jacobian-vector** computations

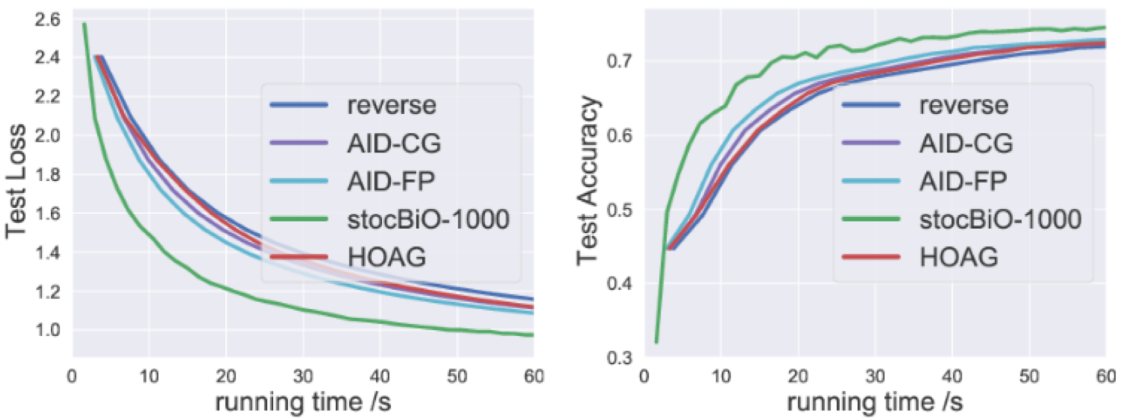
Fast Stochastic Bilevel Optimizer

- Lower complexity

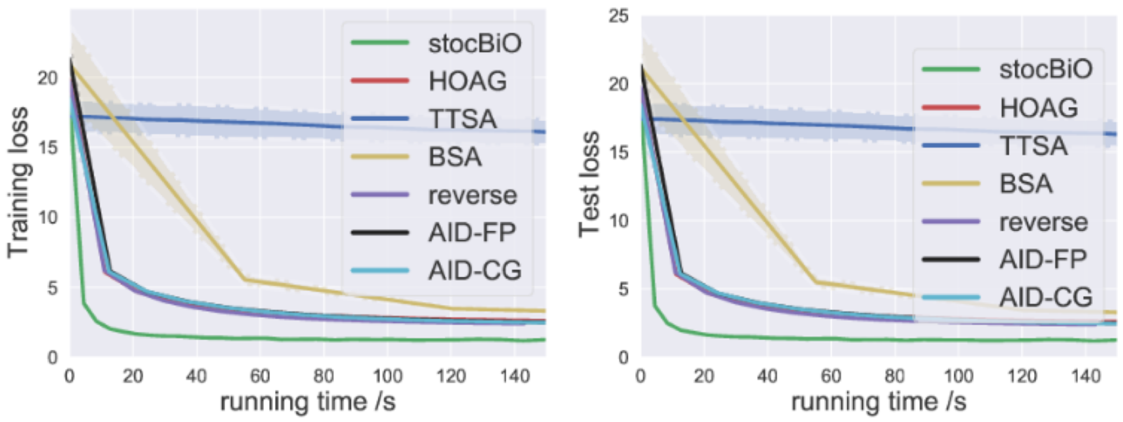
Algorithm	Gc(F,ϵ)	Gc(G,ϵ)	JV(G,ϵ)	HV(G,ϵ)
TTSA (Hong et al., 2020)	$\mathcal{O}(\text{poly}(\kappa)\epsilon^{-\frac{5}{2}})^*$	$\mathcal{O}(\text{poly}(\kappa)\epsilon^{-\frac{5}{2}})$	$\mathcal{O}(\text{poly}(\kappa)\epsilon^{-\frac{5}{2}})$	$\mathcal{O}(\text{poly}(\kappa)\epsilon^{-\frac{5}{2}})$
BSA (Ghadimi & Wang, 2018)	$\mathcal{O}(\kappa^6\epsilon^{-2})$	$\mathcal{O}(\kappa^9\epsilon^{-3})$	$\mathcal{O}\left(\kappa^6\epsilon^{-2}\right)$	$\widetilde{\mathcal{O}}\left(\kappa^6\epsilon^{-2}\right)$
stocBiO (this paper)	$\mathcal{O}(\kappa^5\epsilon^{-2})$	$\mathcal{O}(\kappa^9\epsilon^{-2})$	$\mathcal{O}\left(\kappa^5\epsilon^{-2}\right)$	$\widetilde{\mathcal{O}}\left(\kappa^6\epsilon^{-2}\right)$

ϵ: target accuracy; κ: condition number

- Fast convergence and strong efficiency:



Logistic regression on 20 Newsgroup



Data hyper-cleaning on MNIST

Summary

- New non-asymptotic analysis
  - ➢ Tighter analysis on AID-based bilevel optimizers
  - ➢ First-known analysis on ITD-based bilevel optimizers

- Faster stochastic bilevel algorithm
  - ➢ Lower sample complexity
  - ➢ Better efficiency, scalability and test performance

- Future works
  - ➢ Application to reinforcement learning
  - ➢ Hessian and Jacobian free methods

Thanks!