Optimal Hessian/Jacobian-Free Nonconvex-PL Bilevel Optimization

Feihu Huang

Nanjing University of Aeronautics and Astronautics

[1] Feihu Huang. Optimal Hessian/Jacobian-Free Nonconvex-PL Bilevel Optimization. **ICML**, 2024. **(Oral)**

Outline

- Background
- Existing Algorithms for Nonconvex-PL Bilevel Optimization
- Our Optimal Hessian/Jacobian-Free Algorithm
- Convergence Analysis
- Experimental Results
- Conclusions

Outline

- Background
- Existing Algorithms for Nonconvex-PL Bilevel Optimization
- Our Optimal Hessian/Jacobian-Free Algorithm
- Convergence Analysis
- Experimental Results
- Conclusions

Background

Bilevel optimization can effectively capture the inherent nested structures of problems, thus, it recently has been widely used in many machine learning tasks such as hyper-parameter learning, meta learning and reinforcement learning.

$$\min_{x \in \mathbb{R}^d} \; F(x) \equiv f\big(x, y^*(x)\big)$$

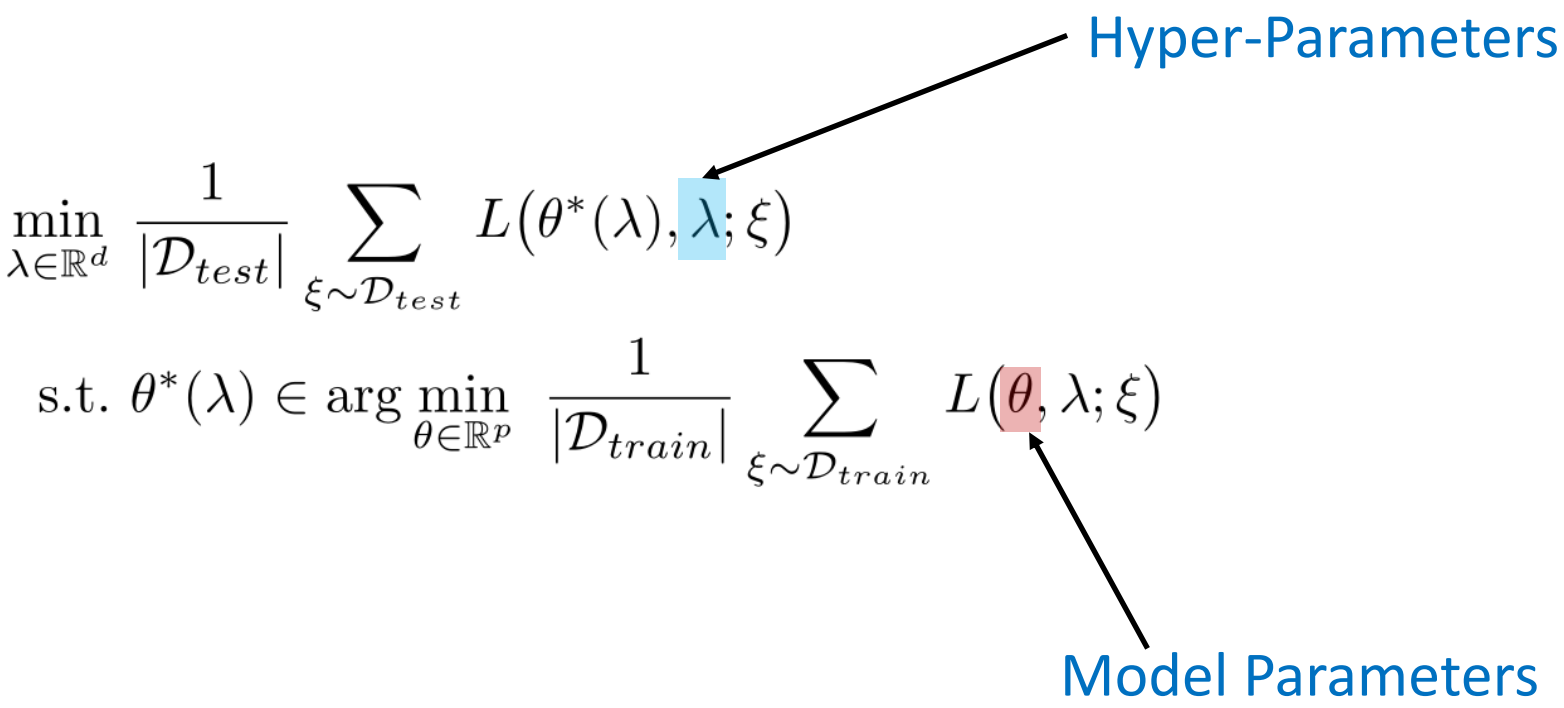
$$\text{s.t.} \; y^*(x) \in \arg\min_{y \in \mathbb{R}^p} \; g(x, y)$$

Upper-Level

Lower-Level

Background

Hyper-Parameter Leaning：



Background

1. When $g(x, y)$ on variable y is strongly convex and twice differentiable,

$$\min_{x \in \mathbb{R}^d} F(x) \equiv f\left(x, y^*(x)\right)$$
$$\text{s.t. } y^*(x) \in \arg\min_{y \in \mathbb{R}^p} g(x, y)$$

$$\nabla F(x) = \nabla_x f(x, y^*(x)) - \nabla^2_{xy} g(x, y^*(x)) \left[\nabla^2_{yy} g(x, y^*(x))\right]^{-1} \nabla_y f(x, y^*(x))$$

Approximate

$$\Phi(x, y) = \nabla_x f(x, y) - \nabla^2_{xy} g(x, y) \left[\nabla^2_{yy} g(x, y)\right]^{-1} \nabla_y f(x, y)$$

Background

1. When $g(x, y)$ on variable y is strongly convex and twice differentiable,

$$\Phi(x,y) = \nabla_x f(x,y) - \nabla^2_{xy} g(x,y) \left[\nabla^2_{yy} g(x,y)\right]^{-1} \nabla_y f(x,y)$$

Approximated Gradient Algorithm [4]

for $k = 0, 1, \ldots, K-1$ **do**
  for $t = 0, 1, \ldots, T-1$ **do**
    $y_k^{t+1} = y_k^t - \gamma \nabla_y g(x_k, y_k^t)$;
  end for
  Set $y_{k+1} = y_k^T$;
  $x_{k+1} = x_k - \lambda \Phi(x_k, y_{k+1})$;
end for

[4] Ghadimi, Saeed, and Mengdi Wang. "Approximation methods for bilevel programming." *arXiv preprint arXiv:1802.02246* (2018).

Background

2. When $g(x, y)$ on variable y is non-strongly-convex,

Bilevel Optimization

$$\min_{x \in \mathbb{R}^d} F(x) \equiv f\left(x, y^*(x)\right)$$
$$\text{s.t. } y^*(x) \in \arg\min_{y \in \mathbb{R}^p} g(x, y)$$

Equivalent

Single-Level Constrained Optimization

$$\min_{x \in \mathbb{R}^d, y \in \mathbb{R}^p} f(x, y)$$
$$\text{s.t. } q(x, y) = g(x, y) - \min_{y \in \mathbb{R}^p} g(x, y) \leq 0$$

Background

2. When $g(x, y)$ on variable y is convex or non-convex,

Penalized Gradient Algorithm [5]

for $k = 0, 1, \ldots, K-1$ **do**
  for $t = 0, 1, \ldots, T-1$ **do**
    $y_k^{t+1} = y_k^t - \gamma \nabla_y g(x_k, y_k^t)$;
  end for
  Set $\hat{q}(x, y) = g(x, y) - g(x, y_k^T)$;
  $x_{k+1} = x_k - \lambda(\nabla_x f(x_k, y_k) + \alpha_k \nabla_x \hat{q}(x_k, y_k))$;
  $y_{k+1} = y_k - \lambda(\nabla_y f(x_k, y_k) + \alpha_k \nabla_y \hat{q}(x_k, y_k))$;
end for

Penalty parameter

[5] Ye, M., Liu, B., Wright, S., Stone, P., and Liu, Q. Bome! bilevel optimization made easy: A simple first-order approach. In NeurIPS, 17248–17262, 2022.
[6] Shen H, Chen T. On penalty-based bilevel gradient descent method, ICML: 30992-31015, 2023.

Outline

- Background
- Existing Algorithms for Nonconvex-PL Bilevel Optimization
- Our Optimal Hessian/Jacobian-Free Algorithm
- Convergence Analysis
- Experimental Results
- Conclusions

Existing Algorithms for Nonconvex-PL Bilevel

In this talk, we force on solving the nonconvex-Polyak-Łojasiewicz (PL) bilevel optimization:

$$\min_{x \in \mathcal{X} \subseteq \mathbb{R}^d} F(x) \equiv f\left(x, y^*(x)\right)$$

$$\text{s.t. } y^*(x) \in \arg\min_{y \in \mathbb{R}^p} g(x, y)$$

nonconvex on x and y

nonconvex- PL on y

Existing Algorithms for Nonconvex-PL Bilevel

Recently, most existing methods build on penalty or Lagrangian methods by solving the following transformed constrained optimization, but they suffer from high convergence (or gradient) complexity.

$$\min_{x\in\mathcal{X}\subseteq\mathbb{R}^d, y\in\mathbb{R}^p} f(x,y)$$
$$\text{s.t. } g(x,y) - \min_{y\in\mathbb{R}^p} g(x,y) \leq 0$$

BOME [5] ,V-PBGD[6], Prox-F2BA [8],SLM [9]

[5]Ye, M., Liu, B., Wright, S., Stone, P., and Liu, Q. Bome! bilevel optimization made easy: A simple first-order approach. In NeurIPS, 17248–17262, 2022.
[6] Shen H, Chen T. On penalty-based bilevel gradient descent method, ICML, 2023: 30992-31015.
[8] Kwon J, Kwon D, Wright S, et al. On penalty methods for nonconvex bilevel optimization and first-order stochastic approximation[J]. arXiv preprint arXiv:2309.01753, 2023.
[9] Lu S. SLM: A Smoothed First-Order Lagrangian Method for Structured Constrained Nonconvex Optimization. Advances in NeurIPS, 2023.

Existing Algorithms for Nonconvex-PL Bilevel

Afterwards, our MGBiO [2] method directly solves the nonconvex-PL bilevel optimization

$$\min_{x\in\mathcal{X}\subseteq\mathbb{R}^d} F(x) \equiv f\big(x, y^*(x)\big)$$

$$\text{s.t. } y^*(x) \in \arg\min_{y\in\mathbb{R}^p} g\big(x, y\big)$$

[2] Huang, Feihu. "On momentum-based gradient methods for bilevel optimization with nonconvex lower-level." arXiv preprint arXiv:2303.03944 , 2023.

Existing Algorithms for Nonconvex-PL Bilevel

Assumption 1. *The function $g(x,\cdot)$ satisfies the μ-Polyak-Łojasiewicz (PL) condition for some $\mu > 0$ if for any given $x \in \mathcal{X}$, it holds that*

$$\|\nabla_y g(x,y)\|^2 \geq 2\mu\big(g(x,y) - \min_y g(x,y)\big), \quad \forall y \in \mathbb{R}^p.$$

reasonable mild

Assumption 2. *The function $g(x,y)$ satisfies*

$$\varrho\big(\nabla^2_{yy} g\big(x, y^*(x)\big)\big) \in [\mu, L_g],$$

where $y^*(x) \in \arg\min_y g(x,y)$, and $\varrho(\cdot)$ denotes the eigenvalue (or singular-value) function and $L_g \geq \mu > 0$.

Support

Lemma B.1. *For a μ-PL function $h(x) : \mathbb{R}^d \to \mathbb{R}$ that is twice differentiable, at any $x^* \in \arg\min_x h(x)$,*

$$\lambda^+_{\min}(\nabla^2 h(x^*)) \geq \mu,$$

where $\lambda^+_{\min}(\cdot)$ denotes the smallest non-zero eigenvalue.

Existing Algorithms for Nonconvex-PL Bilevel

$g(x,y)$ on variable y is non-convex and satisfies PL condition

$$\min_{x\in\mathcal{X}\subseteq\mathbb{R}^d} F(x) \equiv f\big(x, y^*(x)\big)$$
$$\text{s.t. } y^*(x) \in \arg\min_{y\in\mathbb{R}^p} g(x,y)$$

Nonconvex- PL on y

$$\nabla F(x) = \nabla_x f(x, y^*(x)) - \nabla^2_{xy} g(x, y^*(x)) \Big[\nabla^2_{yy} g(x, y^*(x))\Big]^{-1} \nabla_y f(x, y^*(x))$$

Approximate

$$\Phi(x,y) = \nabla_x f(x,y) - \nabla^2_{xy} g(x,y) \big(\mathcal{S}_{[\mu, L_g]}\big[\nabla^2_{yy} g(x,y)\big]\big)^{-1} \nabla_y f(x,y)$$

Existing Algorithms for Nonconvex-PL Bilevel

Algorithm 1 MGBiO Algorithm

Input: T, parameters $\{\gamma, \lambda, \eta_t\}$ and initial input $x_1 \in \mathcal{X}$ and $y_1 \in \mathbb{R}^p$;
for $t = 1, 2, \ldots, T$ **do**
  $v_t = \nabla_y g(x_t, y_t)$, $u_t = \nabla_x f(x_t, y_t)$, $h_t = \nabla_y f(x_t, y_t)$, $G_t = \nabla^2_{xy} g(x_t, y_t)$;
  $H_t = \mathcal{S}_{[\mu, L_g]}\left[\nabla^2_{yy} g(x_t, y_t)\right] = U_t \Theta_t U_t^T$, where $\theta_{t,i} \in [\mu, L_g]$ for all $i = 1, \cdots, p$;
  $w_t = u_t - G_t (H_t)^{-1} h_t = \nabla_x f(x_t, y_t) - G_t\Big(\sum_{i=1}^p \left(U_{t,i}^T h_t\right)/\theta_{t,i} U_{t,i}\Big)$;
  $\tilde{x}_{t+1} = \arg\min_{x \in \mathcal{X}} \left\{\langle w_t, x\rangle + \frac{1}{2\gamma}\|x - x_t\|^2\right\}$ and $x_{t+1} = x_t + \eta_t(\tilde{x}_{t+1} - x_t)$;
  $\tilde{y}_{t+1} = y_t - \lambda v_t$ and $y_{t+1} = y_t + \eta_t(\tilde{y}_{t+1} - y_t)$;
end for
Output: Chosen uniformly random from $\{x_t\}_{t=1}^T$.

It requires compute expensive Hessian/Jacobian matrices and its inverses.

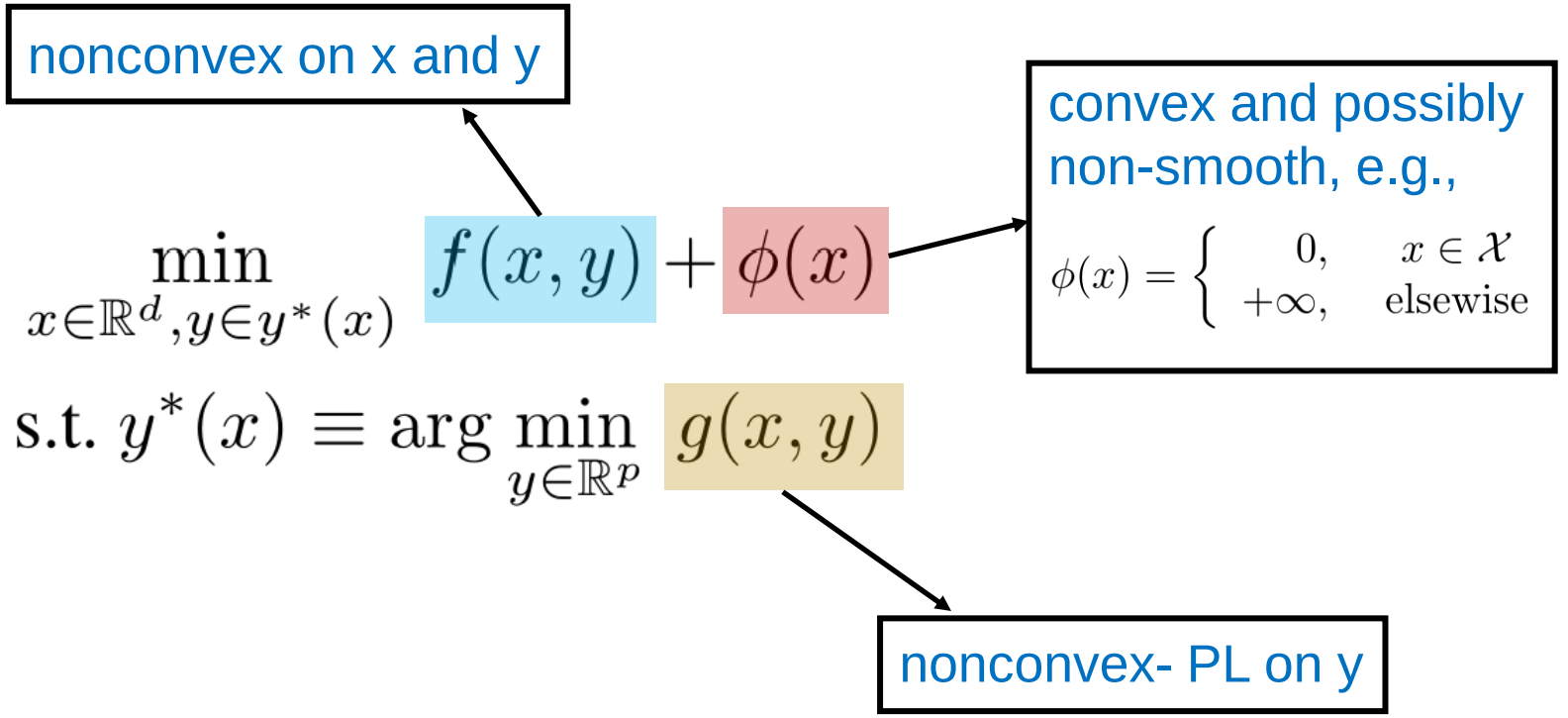
Algorithm	Reference	$g(x,\cdot)$	L.H. on $f(\cdot,\cdot)$	Complexity	Loop(s)	H.J.F.
BOME	(Liu et al., 2022)	PL / local-PL		$O(\epsilon^{-1.5})$ / $O(\epsilon^{-2})$	Double	√
V-PBGD	(Shen & Chen, 2023)	PL / local-PL		$O(\epsilon^{-1.5})$ / $O(\epsilon^{-1.5})$	Double	√
GALET	(Xiao et al., 2023)	SC / PL		$O(\epsilon^{-1})$ / **Meaningless**	Triple	
SLM	(Lu, 2023)	PL / local-PL		$O(\epsilon^{-3.5})$ / $O(\epsilon^{-3.5})$	Double	√
Prox-F^{2}BA	(Kwon et al., 2023)	Proximal-EB	√	$O(\epsilon^{-1.5})$ / $O(\epsilon^{-1.5})$	Double	√
F^{2}BA	(Chen et al., 2024)	PL / local-PL	√	$O(\epsilon^{-1})$ / $O(\epsilon^{-1})$	Double	√
MGBiO	(Huang, 2023b)	PL / local-PL		$O(\epsilon^{-1})$ / $O(\epsilon^{-1})$	Single	
AdaPAG	(Huang, 2023a)	PL / local-PL		$O(\epsilon^{-1})$ / $O(\epsilon^{-1})$	Single	
HJFBiO	Ours	PL / local-PL		$O(\epsilon^{-1})$ / $O(\epsilon^{-1})$	Single	√

Outline

- Background
- Existing Algorithms for Nonconvex-PL Bilevel Optimization
- Our Optimal Hessian/Jacobian-Free Algorithm
- Convergence Analysis
- Experimental Results
- Conclusions

Optimal Hessian/Jacobian-Free Algorithm

In our paper [1], we propose an optimal Hessian/Jacobian-Free method (i.e., HJFBiO) to directly solve the following nonconvex-PL bilevel optimization:



[1] Feihu Huang. Optimal Hessian/Jacobian-Free Nonconvex-PL Bilevel Optimization. **ICML**, 2024.

Optimal Hessian/Jacobian-Free Algorithm

As our MGBiO [2], we have

$$F(x) \equiv f\big(x, y^*(x)\big)$$

$$\nabla F(x) = \nabla_x f(x, y^*(x)) - \nabla^2_{xy} g(x, y^*(x)) \Big[\nabla^2_{yy} g(x, y^*(x))\Big]^{-1} \nabla_y f(x, y^*(x))$$

Approximate

$$\hat{\nabla} f(x, y) = \nabla_x f(x, y) - \nabla^2_{xy} g(x, y) \big(\mathcal{S}_{[\mu, L_g]}\big[\nabla^2_{yy} g(x, y)\big]\big)^{-1} \nabla_y f(x, y)$$

Equivalent

$$\hat{\nabla} f(x, y, v^*) = \nabla_x f(x, y) - \nabla^2_{xy} g(x, y) v^*$$

$$v^* = \arg\min_{v \in \mathbb{R}^p} \left\{ R(x, y, v) \equiv \frac{1}{2} v^T \mathcal{S}_{[\mu, L_g]}\big[\nabla^2_{yy} g(x, y)\big] v - v^T \nabla_y f(x, y) \right\}$$

Optimal Hessian/Jacobian-Free Algorithm

Under this case, we can use the following iterations to solve the nonconvex-PL bilevel problem:

$$\begin{cases} y_{t+1} = y_t - \lambda \nabla_y g(x_t, y_t) \\ x_{t+1} = \mathbb{P}^{\gamma}_{\phi(\cdot)}\left(x_t, \widehat{\nabla} f(x_t, y_t, v_t)\right) \\ v_{t+1} = \mathcal{P}_{r_v}\left(v_t - \tau \nabla_v R(x_t, y_t, v_t)\right) \end{cases}$$

computational complexity

$O(p^3 + pd)$

$$\widehat{\nabla} f(x, y, v) = \nabla_x f(x, y) - \nabla^2_{xy} g(x, y) v$$

$$\nabla_v R(x, y, v) = \mathcal{S}_{[\mu, L_g]}\left[\nabla^2_{yy} g(x, y)\right] v - \nabla_y f(x, y)$$

$$\mathbb{P}^{\gamma}_{\phi(\cdot)}\left(x_t, w_t\right) = \arg\min_{x \in \mathbb{R}^d} \left\{ \langle w_t, x \rangle + \frac{1}{2\gamma} \|x - x_t\|^2 + \phi(x) \right\}$$

$$\mathcal{P}_{r_v}(v) = \{v \in \mathbb{R}^p : \|v\| \leq r_v > 0\}, \; 0 < r_v \leq \frac{C_{fy}}{\mu}$$

Optimal Hessian/Jacobian-Free Algorithm

In our paper, thus, we propose two finite-difference estimators and a new projection operator to avoid computing expensive Hessian/Jacobian matrices and its projection :

$$\widetilde{J}(x, y, v, \delta_\epsilon) = \frac{\nabla_x g(x, y + \delta_\epsilon v) - \nabla_x g(x, y - \delta_\epsilon v)}{2\delta_\epsilon} \longrightarrow \nabla^2_{xy} g(x, y) v$$

Definition 3. *Given matrix* $H \in \mathbb{R}^{p\times p}$ *and vector* $v \in \mathbb{R}^p$, *and* $\mathcal{S}_{[\mu,L_g]}[\cdot]$ *is a projection operator on the set* $\{H \in \mathbb{R}^{p\times p} : \mu \leq \varrho(H) \leq L_g\}$ *where* $\varrho(\cdot)$ *denotes the eigenvalue function, and* $\mathcal{P}_{r_v}(\cdot)$ *is a projection operator onto the set* $\{v \in \mathbb{R}^p : \|v\| \leq r_v\}$, *then we define a* ***new projection operator*** $\mathcal{M}_{r_h}(\cdot,\cdot)$ *on set* $\{H \in \mathbb{R}^{p\times p}, v \in \mathbb{R}^p : \|Hv\| \leq r_h\}$, *which satisfies*

$$\mathcal{M}_{r_h}(H, v) \triangleq \mathcal{S}_{[\mu,L_g]}[H]\mathcal{P}_{r_v}(v),$$

where $0 < r_h \leq r_v L_g$. *For notational simplicity, let* $\mathcal{M}_{r_h}(H, v) = \mathcal{M}_{r_h}(Hv)$.

Optimal Hessian/Jacobian-Free Algorithm

In our paper, thus, we propose two finite-difference estimators and a new projection operator to avoid computing expensive Hessian/Jacobian matrices and its projection :

$$\widetilde{H}(x, y, v, \delta_\epsilon) = \frac{\nabla_y g(x, y + \delta_\epsilon v) - \nabla_y g(x, y - \delta_\epsilon v)}{2\delta_\epsilon} \longrightarrow \nabla^2_{yy} g(x, y) v$$

$$\mathcal{M}_{r_h}\left(\widetilde{H}(x_t, y_t, v_t, \delta_\epsilon)\right) \longrightarrow \mathcal{S}_{[\mu, L_g]}\left[\nabla^2_{yy} g(x, y)\right] v$$

Optimal Hessian/Jacobian-Free Algorithm

Then our HJFBiO method use the following iterations:

$$\begin{cases} y_{t+1} = y_t - \lambda \nabla_y g(x_t, y_t) \\ x_{t+1} = \mathbb{P}^{\gamma}_{\phi(\cdot)}\left(x_t, \widetilde{\nabla} f(x_t, y_t, v_t)\right) \\ v_{t+1} = \mathcal{P}_{r_v}\left(v_t - \tau \widetilde{\nabla}_v R(x_t, y_t, v_t)\right) \end{cases}$$

computational complexity

$O(p+d)$

$$\widetilde{\nabla} f(x_t, y_t, v_t) = \nabla_x f(x_t, y_t) - \widetilde{J}(x_t, y_t, v_t, \delta_\epsilon)$$

$$\widetilde{\nabla}_v R(x_t, y_t, v_t) = \mathcal{M}_{r_h}\left(\widetilde{H}(x_t, y_t, v_t, \delta_\epsilon)\right) - \nabla_y f(x_t, y_t)$$

$$\mathbb{P}^{\gamma}_{\phi(\cdot)}\left(x_t, w_t\right) = \arg\min_{x \in \mathbb{R}^d} \left\{ \langle w_t, x \rangle + \frac{1}{2\gamma} \|x - x_t\|^2 + \phi(x) \right\}$$

$$\mathcal{P}_{r_v}(v) = \{ v \in \mathbb{R}^p : \|v\| \leq r_v > 0 \}, \ 0 < r_v \leq \frac{C_{fy}}{\mu}$$

Optimal Hessian/Jacobian-Free Algorithm

Algorithm 1 Hessian/Jacobian-free Bilevel Optimization (i.e, HJFBiO) Algorithm

1: **Input:** T, learning rates $\lambda > 0$, $\gamma > 0$, $\tau > 0$, and tuning parameters $\delta_\epsilon > 0$, $r_v > 0$, $r_h > 0$, and initial input $x_1 \in \mathbb{R}^d$, $y_1 \in \mathbb{R}^p$ and $v_1 \in \mathbb{R}^p$;

2: **for** $t = 1, 2, \ldots, T$ **do**

3: Compute $u_t = \nabla_y g(x_t, y_t)$, and update $y_{t+1} = y_t - \lambda u_t$;

4: Compute $w_t = \widetilde{\nabla} f(x_t, y_t, v_t) = \nabla_x f(x_t, y_t) - \widetilde{J}(x_t, y_t, v_t, \delta_\epsilon)$, and update $x_{t+1} = \mathbb{P}^{\gamma}_{\phi(\cdot)}\big(x_t, w_t\big)$;

5: Compute $h_t = \widetilde{\nabla}_v R(x_t, y_t, v_t) = \mathcal{M}_{r_h}\big(\widetilde{H}(x_t, y_t, v_t, \delta_\epsilon)\big) - \nabla_y f(x_t, y_t)$, and update $v_{t+1} = \mathcal{P}_{r_v}\big(v_t - \tau h_t\big)$;

6: **end for**

7: **Output:** Chosen uniformly random from $\{x_t\}_{t=1}^T$.

Outline

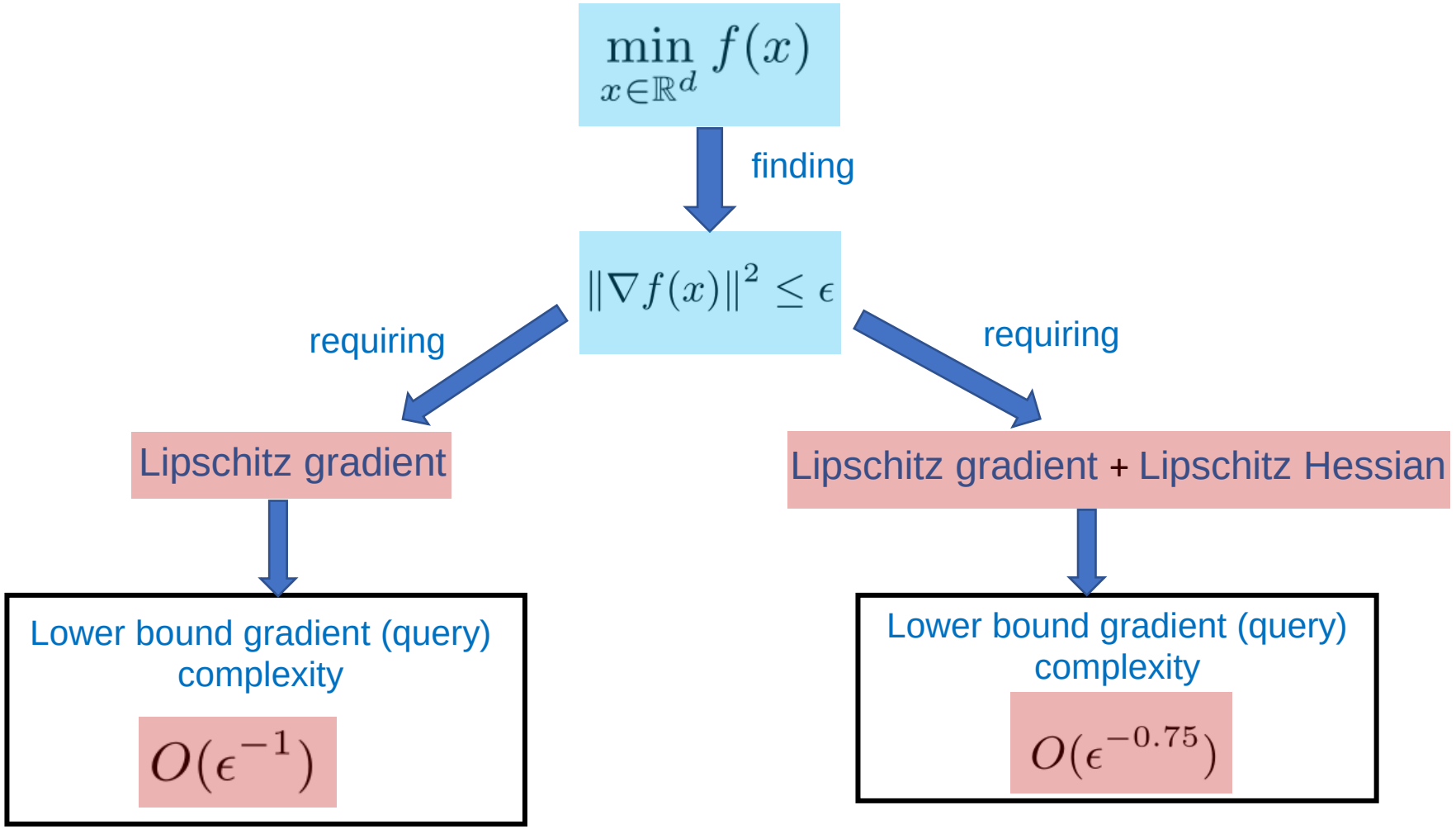
- Background
- Existing Algorithms for Nonconvex-PL Bilevel Optimization
- Our Optimal Hessian/Jacobian-Free Algorithm
- Convergence Analysis
- Experimental Results
- Conclusions

Convergence Analysis

Table 1: Comparison of **gradient (or iteration) complexity** between our method and the existing methods in solving bilevel problem (1) for finding an ϵ-stationary solution ($\|\nabla F(x)\|^2 \leq \epsilon$ or its equivalent variants, where $F(x) = f(x, y)$ with $y \in y^*(x)$). Here $g(x, \cdot)$ denotes function on the second variable y with fixing variable x. **SC** stands for strongly convex. **H.J.F.** stands for Hessian/Jacobian-Free. **L.H.** stands for Lipschitz Hessian condition. *The Prox-F^{2}BA and F^{2}BA methods rely on some strict conditions such as Lipschitz Hessian of function $f(x, y)$.* **Note that** the GALET (Xiao et al., 2023) method simultaneously uses the PL condition, its Assumption 2 (i.e., let $\sigma_g = \inf_{x,y}\{\sigma^+_{\min}(\nabla^2_{yy} g(x, y))\} > 0$ for all (x, y)) and its Assumption 1 (i.e., $\nabla^2_{yy} g(x, y)$ is Lipschitz continuous). Clearly, when Hessian matrix $\nabla^2_{yy} g(x, y)$ is singular, its Assumption 1 and Assumption 2 imply that the lower bound of the non-zero singular values σ_g is close to zero (i.e., $\sigma_g \to 0$), under this case, the convergence results of the GALET are **meaningless**, e.g., the constant $L_w = \frac{\ell_{f,1}}{\sigma_g} + \frac{\sqrt{2}\ell_{g,2}\ell_{f,0}}{\sigma_g^2} \to +\infty$ used in its Lemmas 6 and 9. Under the other case, the PL condition, Lipschitz continuous of Hessian and its Assumption 2 (the singular values of Hessian is bounded away from 0, i.e., $\sigma_g > 0$) imply that GALET assumes strongly convex (Detailed discussion in the Appendix B).

Algorithm	**Reference**	$g(x, \cdot)$	**L.H.** on $f(\cdot, \cdot)$	**Complexity**	**Loop(s)**	**H.J.F.**
BOME	(Liu et al., 2022)	PL / local-PL		$O(\epsilon^{-1.5})$ / $O(\epsilon^{-2})$	Double	$\surd$
V-PBGD	(Shen & Chen, 2023)	PL / local-PL		$O(\epsilon^{-1.5})$ / $O(\epsilon^{-1.5})$	Double	$\surd$
GALET	(Xiao et al., 2023)	SC / PL		$O(\epsilon^{-1})$ / **Meaningless**	Triple	
SLM	(Lu, 2023)	PL / local-PL		$O(\epsilon^{-3.5})$ / $O(\epsilon^{-3.5})$	Double	$\surd$
Prox-F^{2}BA	(Kwon et al., 2023)	Proximal-EB	$\surd$	$O(\epsilon^{-1.5})$ / $O(\epsilon^{-1.5})$	Double	$\surd$
F^{2}BA	(Chen et al., 2024)	PL / local-PL	$\surd$	$O(\epsilon^{-1})$ / $O(\epsilon^{-1})$	Double	$\surd$
MGBiO	(Huang, 2023b)	PL / local-PL		$O(\epsilon^{-1})$ / $O(\epsilon^{-1})$	Single	
AdaPAG	(Huang, 2023a)	PL / local-PL		$O(\epsilon^{-1})$ / $O(\epsilon^{-1})$	Single	
HJFBiO	Ours	PL / local-PL		$O(\epsilon^{-1})$ / $O(\epsilon^{-1})$	Single	$\surd$

Convergence Analysis



[10] Carmon, Y., Duchi, J. C., Hinder, O., and Sidford, A. Lower bounds for finding stationary points i. Mathematical Programming, 184(1-2):71–120, 2020

Convergence Analysis

$\|\nabla F(x)\|^2 \leq \epsilon$ or its equivalent variants, where $F(x) = f(x, y)$ with $y \in y^*(x) = \arg\min_y g(x, y)$.

Algorithm	Reference	$g(x, \cdot)$	L.H. on $f(\cdot, \cdot)$	Complexity	Loop(s)	H.J.F.
BOME	(Liu et al., 2022)	PL / local-PL		$O(\epsilon^{-1.5})$ / $O(\epsilon^{-2})$	Double	√
V-PBGD	(Shen & Chen, 2023)	PL / local-PL		$O(\epsilon^{-1.5})$ / $O(\epsilon^{-1.5})$	Double	√
GALET	(Xiao et al., 2023)	SC / PL		$O(\epsilon^{-1})$ / **Meaningless**	Triple	
SLM	(Lu, 2023)	PL / local-PL		$O(\epsilon^{-3.5})$ / $O(\epsilon^{-3.5})$	Double	√
Prox-F^{2}BA	(Kwon et al., 2023)	Proximal-EB	√	$O(\epsilon^{-1.5})$ / $O(\epsilon^{-1.5})$	Double	√
F^{2}BA	(Chen et al., 2024)	PL / local-PL	√	$O(\epsilon^{-1})$ / $O(\epsilon^{-1})$	Double	√
MGBiO	(Huang, 2023b)	PL / local-PL		$O(\epsilon^{-1})$ / $O(\epsilon^{-1})$	Single	
AdaPAG	(Huang, 2023a)	PL / local-PL		$O(\epsilon^{-1})$ / $O(\epsilon^{-1})$	Single	
HJFBiO	Ours	PL / local-PL		$O(\epsilon^{-1})$ / $O(\epsilon^{-1})$	Single	√

Only uses full first-order information, and only assume the upper-level objective function $f(x, y)$ has Lipschitz gradient.

Optimal

Outline

- Background
- Existing Algorithms for Nonconvex-PL Bilevel Optimization
- Our Optimal Hessian/Jacobian-Free Algorithm
- Convergence Analysis
- Experimental Results
- Conclusions

Experimental Results

1) Bilevel Polyak-Łojasiewicz Game

$$\min_{x\in\mathbb{R}^d} \ \frac{1}{2}x^T Px + x^T R^1 y,$$
$$\text{s.t.} \ \min_{y\in\mathbb{R}^d} \frac{1}{2}y^T Qy + x^T R^2 y,$$

where $P = \frac{1}{n}\sum_{i=1}^n p_i(p_i)^T$, $Q = \frac{1}{n}\sum_{i=1}^n q_i(q_i)^T$, $R^1 = \frac{1}{n}\sum_{i=1}^n 0.01 r_i^1(r_i^1)^T$ and $R^2 = \frac{1}{n}\sum_{i=1}^n 0.01 r_i^2(r_i^2)^T$. Here the samples $\{p_i\}_{i=1}^n$, $\{q_i\}_{i=1}^n$, $\{r_i^1\}_{i=1}^n$ and $\{r_i^2\}_{i=1}^n$ are independently drawn from Gaussian distributions $\mathcal{N}(0,\Sigma_P)$, $\mathcal{N}(0,\Sigma_Q)$, $\mathcal{N}(0,\Sigma_{R^1})$ and $\mathcal{N}(0,\Sigma_{R^2})$, respectively. Meanwhile, we set $\Sigma_P = U^1D^1(U^1)^T$, where $U^1 \in \mathbb{R}^{d\times l}$ $(l < d)$ is column orthogonal, and $D^1 \in \mathbb{R}^{l\times l}$ is diagonal and its diagonal elements are distributed uniformly in the interval $[\mu, L]$ with $0 < \mu < L$. Let $\Sigma_Q = U^2D^2(U^2)^T$, where $U^2 \in \mathbb{R}^{d\times l}$ is column orthogonal, and $D^2 \in \mathbb{R}^{l\times l}$ is diagonal and its diagonal elements are distributed uniformly in the interval $[\mu, L]$ with $0 < \mu < L$. We also set $\Sigma_{R^1} = 0.001V^1(V^1)^T$ and $\Sigma_{R^2} = 0.001V^2(V^2)^T$, where each element of $V^1, V^2 \in \mathbb{R}^{d\times d}$ is independently sampled from normal distribution $\mathcal{N}(0,1)$. Since the covariance matrices Σ_P and Σ_Q are rank-deficient, it is ensured that both P and Q are singular. Hence the lower-level and upper-level objective functions may be not convex, while they satisfy the PL condition. In this experiment, we set $d = 50$, $l = 48$ and $n = 2500$. For fair comparison, we set the basic learning rate as 0.01

Experimental Results

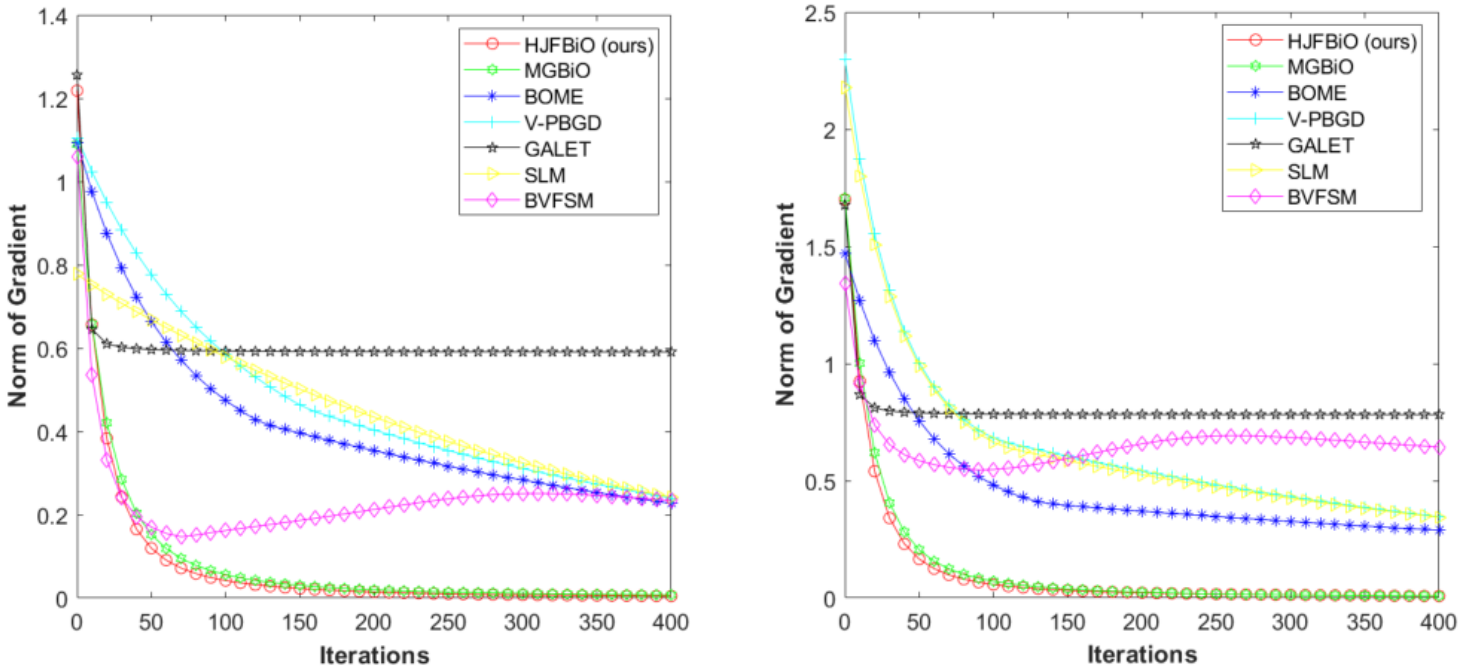


Figure 1: PL Game: norm of gradient vs number of iteration under $d = 100$ (Left) and $d = 200$ (Right).

Experimental Results

2) Hyper-Representation Learning

methods. Learning hyper-representation is one of key points of meta learning, which extract better feature representations to be applied to many different tasks. Here we specifically consider the hyper-representation learning in matrix sensing task. Given n sensing matrices $\{C_i \in \mathbb{R}^{d\times d}\}_{i=1}^n$ with n observations $e_i = \langle C_i, H^* \rangle = \text{trace}(C_i^T H^*)$, where $H^* = U^*(U^*)^T$ is a low-rank symmetric matrix with $U^* \in \mathbb{R}^{d\times r}$. The goal of matrix sensing task is to find the matrix U^*, which can be represented the following problem:

$$\min_{U\in\mathbb{R}^{d\times r}} \frac{1}{n}\sum_{i=1}^{n} \ell_i(U) = \frac{1}{2}\left(\langle C_i, UU^T\rangle - e_i\right)^2.$$

Experimental Results

Then we consider the hyper-representation learning in matrix sensing task, which be rewritten the following bilevel optimization problem:

$$\min_{x\in\mathbb{R}^{d\times r-1}} \frac{1}{|D_v|}\sum_{i\in D_v}\ell_i(x,y^*(x)), \tag{35}$$
$$\text{s.t. } y^*(x)\in\arg\min_{y\in\mathbb{R}^d}\frac{1}{|D_t|}\sum_{i\in D_t}\ell_i(x,y),$$

where $U=[y;x]\in\mathbb{R}^{d\times r}$ is a concatenation of x and y. Here we define variable x to be the first $r-1$ columns of U and variable y to be the last column. Meanwhile, D_t denotes the training dataset, and D_v denotes the validation dataset. The ground truth low-rank matrix H^* is generated by $H^*=U^*(U^*)^T$, where each entry of U^* is drawn from normal distribution $\mathbb{N}(0,\frac{1}{d})$ independently. We randomly generate $n=20d$ samples of sensing matrices $\{C_i\}_{i=1}^n$ from standard normal distribution, and then compute the corresponding no-noise labels $e_i=\langle C_i,H^*\rangle$. We split all samples into two dataset: a train dataset D_t with 40% data and a validation dataset D_v with 60% data.

Experimental Results

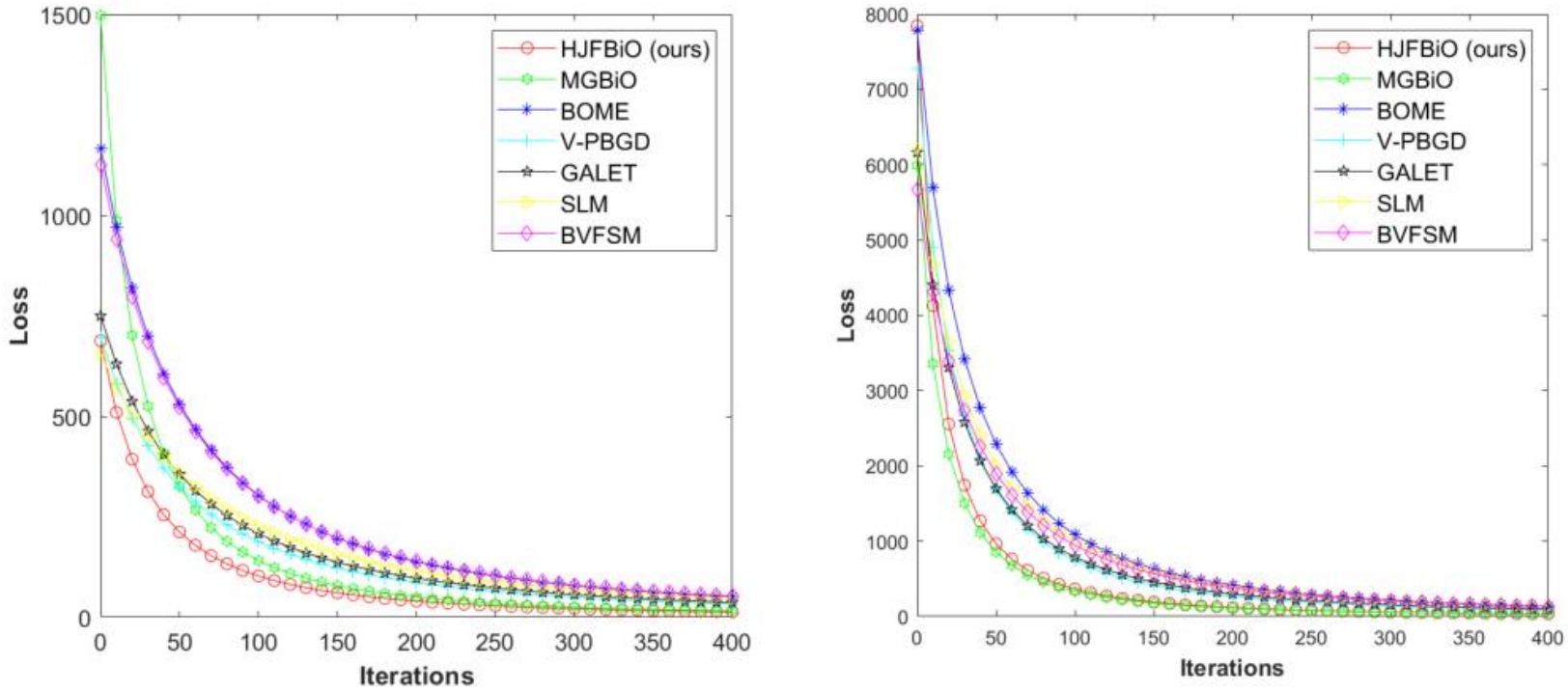


Figure 2: Losses of the algorithms under the case of $d = 100$ (Left) and $d = 200$ (Right).

Outline

- Background
- Existing Algorithms for Nonconvex-PL Bilevel Optimization
- Our Optimal Hessian/Jacobian-Free Algorithm
- Convergence Analysis
- Experimental Results
- Conclusions

Conclusions

1) We proposed an optimal Hessian/Jacobian-Free method for nonconvex-PL Bilevel optimization.

2) We proved our HJFBiO method obtain an optimal gradient complexity for finding stationary solutions of nonconvex-PL Bilevel optimization.

References

[1] Feihu Huang. Optimal Hessian/Jacobian-Free Nonconvex-PL Bilevel Optimization. ICML, 2024.

[2] Feihu Huang. "On momentum-based gradient methods for bilevel optimization with nonconvex lower-level." arXiv preprint arXiv:2303.03944 , 2023.

[3] Feihu Huang. "Adaptive Mirror Descent Bilevel Optimization." arXiv preprint arXiv:2311.04520, 2023.

[4] Ghadimi, Saeed, and Mengdi Wang. "Approximation methods for bilevel programming." *arXiv preprint arXiv:1802.02246* (2018).

[5] Ye, M., Liu, B., Wright, S., Stone, P., and Liu, Q. Bome! bilevel optimization made easy: A simple first-order approach. In NeurIPS, 17248–17262, 2022.

Thanks!

Q&A