An Adaptive Algorithm for Bilevel Optimization on Riemannian Manifolds

Xu Shi* Rufeng Xiao* Rujun Jiang

School of Data Science
Fudan University

NeurIPS 2025

The Problem Formulation

What is RBO?

- The bilevel optimization problem with variables lying on manifolds $(\mathcal{M}_x, \mathcal{M}_y)$.
- **Formulation:**

$$\begin{aligned} \min_{x \in \mathcal{M}_x} \quad & F(x) := f(x, y^*(x)), \\ \text{s.t.} \quad & y^*(x) = \arg\min_{y \in \mathcal{M}_y} g(x, y), \end{aligned} \tag{1}$$

- **Applications:** Riemannian meta-learning, hyperparameter optimization, low-rank adaptation.

Comparisons with Related Works

Table 1: Comparisons of first-order and second-order complexities for reaching an ϵ-stationary point.

Methods	Space	Adaptive	Type	G_f	G_g	JV_g	HV_g
D-TFBO	Euc	✓	Det	$\mathcal{O}(1/\epsilon)$	$\mathcal{O}(1/\epsilon^2)$	$\mathcal{O}(1/\epsilon)$	$\mathcal{O}(1/\epsilon^2)$
S-TFBO				$\tilde{\mathcal{O}}(1/\epsilon)$	$\tilde{\mathcal{O}}(1/\epsilon)$	$\tilde{\mathcal{O}}(1/\epsilon)$	$\tilde{\mathcal{O}}(1/\epsilon)$
RHGD-HINV	Rie	✗	Det	$\mathcal{O}(1/\epsilon)$	$\tilde{\mathcal{O}}(1/\epsilon)$	$\mathcal{O}(1/\epsilon)$	NA
RHGD-CG				$\mathcal{O}(1/\epsilon)$	$\tilde{\mathcal{O}}(1/\epsilon)$	$\mathcal{O}(1/\epsilon)$	$\tilde{\mathcal{O}}(1/\epsilon)$
RHGD-NS				$\mathcal{O}(1/\epsilon)$	$\tilde{\mathcal{O}}(1/\epsilon)$	$\mathcal{O}(1/\epsilon)$	$\tilde{\mathcal{O}}(1/\epsilon)$
RHGD-AD				$\mathcal{O}(1/\epsilon)$	$\tilde{\mathcal{O}}(1/\epsilon)$	$\mathcal{O}(1/\epsilon)$	$\tilde{\mathcal{O}}(1/\epsilon)$
RSHGD-HINV			F-S	$\mathcal{O}(1/\epsilon^2)$	$\tilde{\mathcal{O}}(1/\epsilon^2)$	$\mathcal{O}(1/\epsilon^2)$	NA
RieBO	Rie	✗	Det	$\mathcal{O}(1/\epsilon)$	$\tilde{\mathcal{O}}(1/\epsilon)$	$\mathcal{O}(1/\epsilon)$	$\tilde{\mathcal{O}}(1/\epsilon)$
RieSBO			Sto	$\mathcal{O}(1/\epsilon^2)$	$\tilde{\mathcal{O}}(1/\epsilon^2)$	$\mathcal{O}(1/\epsilon^2)$	$\tilde{\mathcal{O}}(1/\epsilon^2)$
RF^{2}SA	Rie	✗	Det	$\tilde{\mathcal{O}}(1/\epsilon^{3/2})$	$\tilde{\mathcal{O}}(1/\epsilon^{3/2})$	NA	NA
RF^{2}SA			Sto	$\tilde{\mathcal{O}}(1/\epsilon^{7/2})$	$\tilde{\mathcal{O}}(1/\epsilon^{7/2})$	NA	NA
AdaRHD-GD (Ours)	Rie	✓	Det	$\mathcal{O}(1/\epsilon)$	$\mathcal{O}(1/\epsilon^2)$	$\mathcal{O}(1/\epsilon)$	$\mathcal{O}(1/\epsilon^2)$
AdaRHD-CG (Ours)				$\mathcal{O}(1/\epsilon)$	$\mathcal{O}(1/\epsilon^2)$	$\mathcal{O}(1/\epsilon)$	$\tilde{\mathcal{O}}(1/\epsilon)$

Adaptive Riemannian Hypergradient Descent

Main Idea: A fully adaptive step size strategy that requires **no** prior parameter knowledge.
Strategy: Adapting the step sizes based on accumulated Riemannian (hyper)gradient norms.

Adaptive Riemannian Hypergradient Descent

Main Idea: A fully adaptive step size strategy that requires **no** prior parameter knowledge.

Strategy: Adapting the step sizes based on accumulated Riemannian (hyper)gradient norms.

Inner Loops (Find $\hat{y}_t$ and $\hat{v}_t$, $\epsilon_y = \epsilon_v = \frac{1}{T}$):

- **While** $\|\mathcal{G}_y g(x_t, y_t^k)\|^2_{y_t^k} > \epsilon_y$:

$$b^2_{k+1} = b^2_k + \|\mathcal{G}_y g(x_t, y_t^k)\|^2_{y_t^k}, \quad y_t^{k+1} = \mathrm{Exp}_{y_t^k}(-\frac{1}{b_{k+1}} \mathcal{G}_y g(x_t, y_t^k)).$$

Adaptive Riemannian Hypergradient Descent

Main Idea: A fully adaptive step size strategy that requires **no** prior parameter knowledge.
Strategy: Adapting the step sizes based on accumulated Riemannian (hyper)gradient norms.
Inner Loops (Find $\hat{y}_t$ and $\hat{v}_t$, $\epsilon_y = \epsilon_v = \frac{1}{T}$):

- **While** $\|\mathcal{G}_y g(x_t, y_t^k)\|^2_{y_t^k} > \epsilon_y$:

$$b_{k+1}^2 = b_k^2 + \|\mathcal{G}_y g(x_t, y_t^k)\|^2_{y_t^k}, \quad y_t^{k+1} = \mathrm{Exp}_{y_t^k}(-\frac{1}{b_{k+1}} \mathcal{G}_y g(x_t, y_t^k)).$$

- **While** $\|\nabla_v R(x_t, y_t^{K_t}, v_t^n)\|^2_{y_t^{K_t}} > \epsilon_v$:

$$c_{n+1}^2 = c_n^2 + \|\nabla_v R(x_t, \hat{y}_t, v_t^n)\|^2_{\hat{y}_t},$$

$$v_t^{n+1} = v_t^n - \frac{1}{c_{n+1}} \nabla_v R(x_t, \hat{y}_t, v_t^n).$$

Or $\hat{v}_t = \mathrm{TSCG}(\mathcal{H}_y g(x_t, y_t^{K_t}), \mathcal{G}_y f(x_t, y_t^{K_t}), v_t^0, \epsilon_v)$.

Adaptive Riemannian Hypergradient Descent

Main Idea: A fully adaptive step size strategy that requires **no** prior parameter knowledge.
Strategy: Adapting the step sizes based on accumulated Riemannian (hyper)gradient norms.
Outer Loop (Update x):

- Compute approximate hypergradient:

$$\widehat{\mathcal{G}}F(x_t, y_t^{K_t}, v_t^{N_t}) = \mathcal{G}_x f(x_t, y_t^{K_t}) - \mathcal{G}_{xy}^2 g(x_t, y_t^{K_t})[v_t^{N_t}],$$

- Update adaptive term: $a_{t+1}^2 = a_t^2 + \|\widehat{\mathcal{G}}F(x_t, y_t^{K_t}, v_t^{N_t})\|^2$
- Update x: $x_{t+1} = \mathrm{Exp}_{x_t}(-\frac{1}{a_{t+1}}\widehat{\mathcal{G}}F(x_t, y_t^{K_t}, v_t^{N_t}))$

Adaptive Riemannian Hypergradient Descent

Main Idea: A fully adaptive step size strategy that requires **no** prior parameter knowledge.
Strategy: Adapting the step sizes based on accumulated Riemannian (hyper)gradient norms.
Outer Loop (Update x):

- Compute approximate hypergradient:

$$\widehat{\mathcal{G}}F(x_t, y_t^{K_t}, v_t^{N_t}) = \mathcal{G}_x f(x_t, y_t^{K_t}) - \mathcal{G}_{xy}^2 g(x_t, y_t^{K_t})[v_t^{N_t}],$$

- Update adaptive term: $a_{t+1}^2 = a_t^2 + \|\widehat{\mathcal{G}}F(x_t, y_t^{K_t}, v_t^{N_t})\|^2$
- Update x: $x_{t+1} = \mathrm{Exp}_{x_t}(-\frac{1}{a_{t+1}}\widehat{\mathcal{G}}F(x_t, y_t^{K_t}, v_t^{N_t}))$

AdaRHD is the first fully adaptive, parameter-free algorithm for RBO.

Theoretical Guarantees

Theorem 1

The sequence $\{x_t\}_{t=0}^{T}$ *generated by AdaRHD satisfies,*

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\mathcal{G}F(x_t)\|_{x_t}^2 \leq \frac{C}{T} = \mathcal{O}\left(\frac{1}{T}\right),$$

where C *is a constant.*

Theoretical Guarantees

Theorem 1

The sequence $\{x_t\}_{t=0}^{T}$ generated by AdaRHD satisfies,

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\mathcal{G}F(x_t)\|_{x_t}^2 \leq \frac{C}{T} = \mathcal{O}\left(\frac{1}{T}\right),$$

where C is a constant.

Why is this important?

- This complexity **matches** the rate of existing **non-adaptive** methods.
- We achieve that:
  1. **Practicality:** No need to know or tune parameters carefully.
  2. **Efficiency:** Same theoretical convergence speed.
- The same guarantee holds when using computationally cheaper **retraction mappings** instead of exponential mappings.

Experiments: Demo Problem

$$\begin{aligned} \max_{\mathbf{W}\in \mathrm{St}(d,r)} \quad & \mathrm{trace}(\mathbf{M}^*(\mathbf{W})\mathbf{X}^\top \mathbf{Y}\mathbf{W}^\top) \\ \text{s.t.} \quad & \mathbf{M}^*(\mathbf{W}) = \arg\min_{\mathbf{M}\in \mathbb{S}^d_{++}} \ \langle \mathbf{M}, \mathbf{X}^\top \mathbf{X}\rangle + \langle \mathbf{M}^{-1}, \mathbf{W}\mathbf{Y}^\top \mathbf{Y}\mathbf{W}^\top + \lambda \mathbf{I}\rangle. \end{aligned}$$

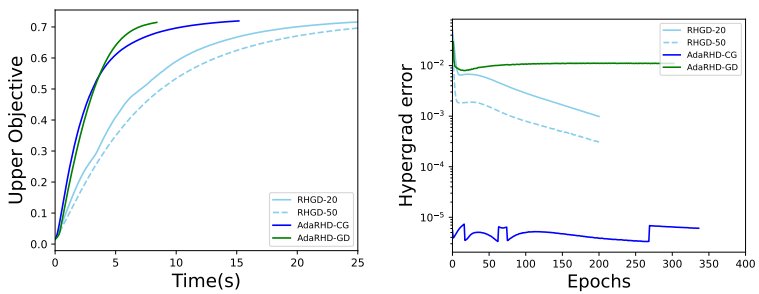


Figure 1: Performances of methods in $n = 100$.

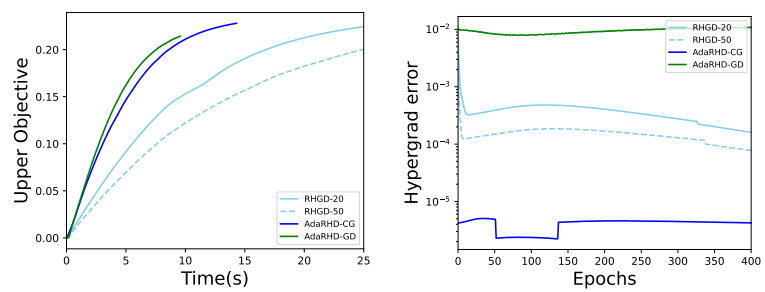


Figure 2: Performances of methods in $n = 1000$.

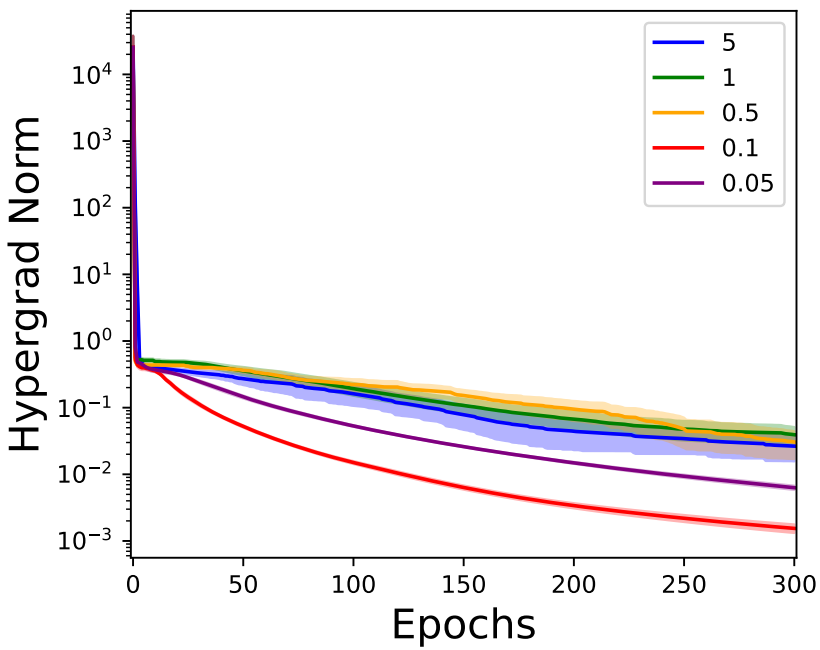
Experiments: Robustness is Key

$$\min_{\mathbf{A}_1,\ \mathbf{A}_2,\ \mathbf{A}_3} \quad -\sum_{i\in\mathcal{D}_{\text{val}}} \frac{\mathbf{y}_i^\top \log(\text{SPDnet}(\mathbf{D}_i;\mathbf{A}_1,\mathbf{A}_2,\mathbf{A}_3)\beta^*(\mathbf{A}_1,\mathbf{A}_2,\mathbf{A}_3))}{|\mathcal{D}_{\text{val}}|},$$

$$\text{s.t.} \quad \beta^*(\mathbf{A}_1,\mathbf{A}_2,\mathbf{A}_3) = \arg\min_{\beta\in\mathbb{R}^{r(r+1)/2}} -\sum_{i\in\mathcal{D}_{\text{tr}}} \frac{\mathbf{y}_i^\top \log(\text{SPDnet}(\mathbf{D}_i;\mathbf{A}_1,\mathbf{A}_2,\mathbf{A}_3)\beta)}{|\mathcal{D}_{\text{tr}}|} + \frac{\lambda}{2}\|\beta\|^2,$$

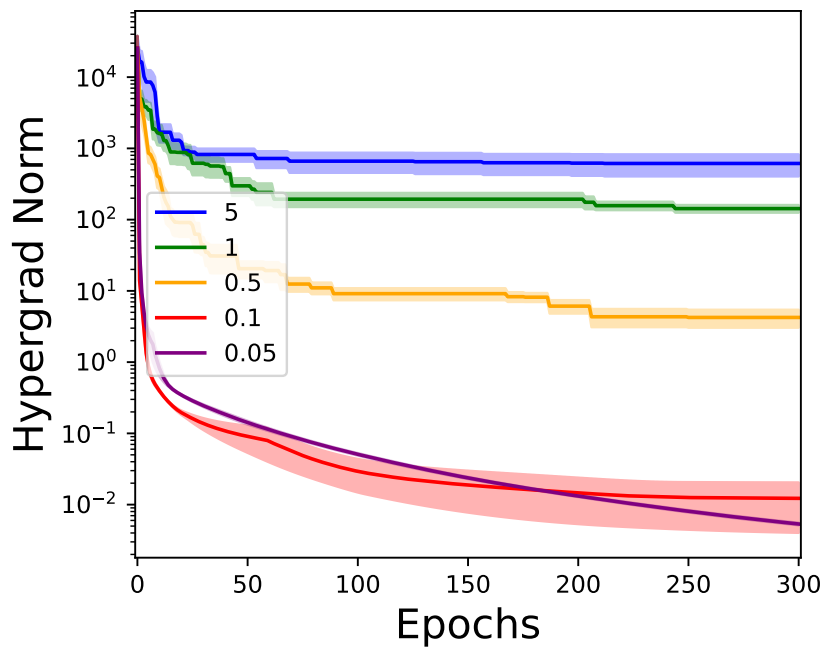
$$\mathbf{A}_1 \in \text{St}(d,d_1),\ \mathbf{A}_2 \in \text{St}(d_1,d_2),\ \mathbf{A}_3 \in \text{St}(d_2,r),$$

Ours (AdaRHD-CG)



Converges for all step sizes.

Baseline (RHGD-20)



Fails for large step sizes.

Thank You!