Lower Bounds of Uniform Stability in Gradient-Based Bilevel Algorithms for Hyperparameter Optimization

Rongzhen Wang[1], Chenyu Zheng[1], Guoqiang Wu[2], Xu Min[3], Xiaolu Zhang[3], Jun Zhou[3], Chongxuan Li[1]

[1]Gaoling School of AI, Renmin University of China

[2]School of Software, Shandong University, [3]Ant Group

NerIPS 2024

Outline

TL;DR

We establish the first **uniform stability lower bounds** for **gradient-based bilevel HO algorithms**, and specifically for the UD-based algorithm, our result verifies the **tightness** of its existing upper bound.

1. Background
   - Hyperparameter optimization (HO)
   - Gradient-based bilevel HO algorithms
   - Stability and generalization in HO
2. Main results
   - Stability lower bounds for UD-based algorithm
   - Construction of the lower bound

Hyperparameter optimization (HO)

- Hyperparameter
  - e.g., regularization coefficient, network topology, feature extractor...
  - specified as input in the **training phase**, optimized in the **validation phase**, and expected to perform well in the **testing phase**

Hyperparameter optimization (HO)

- Hyperparameter
  - e.g., regularization coefficient, network topology, feature extractor...
  - specified as input in the **training phase**, optimized in the **validation phase**, and expected to perform well in the **testing phase**
- Gradient-based HO
  - classical HO (e.g., grid search) can not apply to a large-scale problem
  - optimize $10^4 \sim 10^6$-dimensional hyperparameters
  - applications: feature learning [1], differentiable neural architecture search [2], data reweighting and distillation [3]

Gradient-based bilevel HO algorithms

Let $\boldsymbol{\lambda}$ denote the hyperparameter, and $\boldsymbol{\theta}$ denote the model parameter. Given validation loss $\ell^{\text{val}}(\boldsymbol{\lambda}, \boldsymbol{\theta})$ and inner output $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$, denote that

- compound validation loss: $\mathcal{L}(\boldsymbol{\lambda}) := \ell^{\text{val}}(\boldsymbol{\lambda}, \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}))$, and
- hypergradient: $\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda}) = \nabla_{\boldsymbol{\lambda}}\ell^{\text{val}}\boldsymbol{\lambda}, \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})) + \nabla_{\boldsymbol{\lambda}}\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})\nabla_{\boldsymbol{\theta}}\ell^{\text{val}}\boldsymbol{\lambda}, \hat{\boldsymbol{\theta}}(\boldsymbol{\lambda}))$

Algorithm (Gradient-based bilevel HO, informal)

- **Outer level:** Given optimized $\hat{\boldsymbol{\theta}}(\boldsymbol{\lambda})$, update $\boldsymbol{\lambda}$ by 1-step SGD on S^{val} with hypergradient
  - **Inner level:** Given current $\boldsymbol{\lambda}$, update $\boldsymbol{\theta}$ by K-step SGD on S^{tr}
- Repeat for T steps

UD and IFT-based HO algorithms

- UD: exactly calculate $\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda})$ by *unrolling* the inner *differentiation*
- IFT: approximate $\nabla_{\boldsymbol{\lambda}}\mathcal{L}(\boldsymbol{\lambda})$ by the *implicit function theorem*

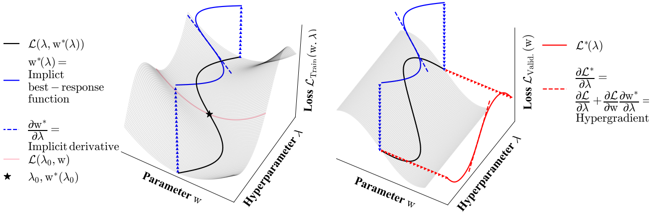

Figure 1.1: Overview of gradient-based HO [3]

Can we estimate the expected testing risk based on the empirical validation risk for the output of an HO algorithm?

Can we estimate the expected testing risk based on the empirical validation risk for the output of an HO algorithm?

Notations

- Data space Z on a target distribution $\mathcal{D}$
- Two i.i.d. samples S^{val} of size m and S^{tr} of size n
- Output hyperparameter $\mathcal{A}(S^{\text{val}}, S^{\text{tr}})$ of an HO algorithm $\mathcal{A}$
- Expected risk of $\boldsymbol{\lambda}$: $R(\boldsymbol{\lambda}) = \mathbb{E}_{\boldsymbol{z}\sim\mathcal{D}}[\mathcal{L}(\boldsymbol{\lambda}; \boldsymbol{z})]$
- Empirical risk of $\boldsymbol{\lambda}$ on S^{val}: $R_{S^{\text{val}}}(\boldsymbol{\lambda}) := \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\boldsymbol{\lambda}; \boldsymbol{z}_i^{\text{val}})$
- **Generalization error:**

$$\epsilon_{\text{gen}} := \mathbb{E}_{\mathcal{A},S^{\text{val}},S^{\text{tr}}}\left[R(\mathcal{A}(S^{\text{val}}, S^{\text{tr}})) - R_{S^{\text{val}}}(\mathcal{A}(S^{\text{val}}, S^{\text{tr}}))\right]$$

Stability and generalization in HO

Uniform stability: the change in the model output when a single validation example is replaced

- Twin validation sets differing at a single example $S^{\text{val}} \simeq \tilde{S}^{\text{val}}$
- $\epsilon_{\text{stab}} :=$ $\sup_{S^{\text{val}} \simeq \tilde{S}^{\text{val}}, S^{\text{tr}}} \mathbb{E}_{\mathcal{A}}[\mathcal{L}(\mathcal{A}(S^{\text{val}}, S^{\text{tr}}); \boldsymbol{z}_i^{\text{val}}) - \mathcal{L}(\mathcal{A}(\tilde{S}^{\text{val}}, S^{\text{tr}}); \tilde{\boldsymbol{z}}_i^{\text{val}})]$
- $\epsilon_{\text{arg}} := \sup_{S^{\text{val}} \simeq \tilde{S}^{\text{val}}, S^{\text{tr}}} \mathbb{E}_{\mathcal{A}}[\|\mathcal{A}(S^{\text{val}}, S^{\text{tr}}) - \mathcal{A}(\tilde{S}^{\text{val}}, S^{\text{tr}})\|]$

Theorem 1.1 (Generalization bound via uniform stability, [4])

For HO algorithms, uniform stability guarantees generalization, i.e., $\epsilon_{\text{gen}} \leq \epsilon_{\text{stab}}$, and if the compound validation loss $\mathcal{L}$ is L-Lipschitz continuous, we have $\epsilon_{\text{gen}} \leq L\epsilon_{\text{arg}}$.

Existing stability upper bound

Theorem 1.2 (Stability upper bound for UD-based algorithm, [4])

Suppose in an HO problem, ℓ^{val} is second order continuously differentiable, ℓ^{tr} is third order continuously differentiable, and ℓ^{tr} is γ^{tr}-smooth w.r.t. $\boldsymbol{\theta}$. Then, solving it with UD-based HO algorithm leads to a L-Lipschitz continuous and γ-smooth compound validation loss $\mathcal{L}$ where $L \lesssim (1+\eta\gamma^{\mathrm{tr}})^K$, $\gamma \lesssim (1+\eta\gamma^{\mathrm{tr}})^{2K}$ and uniform argument stability that

$$\epsilon_{\mathrm{arg}} \leq \sum_{t=1}^{T} \prod_{s=t+1}^{T+1} \left(1+\alpha_s(1-1/m)\gamma\right) \frac{2\alpha_t L}{m}.$$

Tightness of this stability upper bound?

Stability lower bounds for UD-based algorithm

Theorem 2.1 (Stability lower bound for UD-based algorithm)

There exists an HO problem where ℓ^{val} is second order continuously differentiable, ℓ^{tr} is third order continuously differentiable, and ℓ^{tr} is γ^{tr}-smooth w.r.t. $\boldsymbol{\theta}$, such that solving it with UD-based HO algorithm has uniform argument stability that

$$\epsilon_{\mathrm{arg}} \geq \sum_{t=1}^{T} \prod_{s=t+1}^{T+1} \left(1 + \alpha_s (1 - 1/m) \gamma'\right) \frac{2\alpha_t L'}{m},$$

where $L' \asymp L \asymp (1 + \eta\gamma^{\mathrm{tr}})^K$, $\gamma' = \gamma \asymp (1 + \eta\gamma^{\mathrm{tr}})^{2K}$. Here L and γ denote the Lipschitz continuous and smooth coefficients of $\mathcal{L}$.

Stability lower bounds for UD-based algorithm

1. For constant step sizes (i.e., $\alpha_t = c$),

$$\epsilon_{\text{arg}} \asymp \frac{\left(1+c(1-1/m)\gamma\right)^T}{m}.$$

2. For linearly decreasing step sizes (i.e., $\alpha_t \leq c/t$), with additional scaling steps,

$$\frac{T^{\ln\left(1+(1-\frac{1}{m})c\gamma\right)}}{m} \lesssim \epsilon_{\text{arg}} \lesssim \frac{T^{(1-\frac{1}{m})c\gamma}}{m}.$$

Stability lower bounds for UD-based algorithm

1. For constant step sizes (i.e., $\alpha_t = c$),

$$\epsilon_{\text{arg}} \asymp \frac{\left(1+c(1-1/m)\gamma\right)^T}{m}.$$

2. For linearly decreasing step sizes (i.e., $\alpha_t \le c/t$), with additional scaling steps,

$$\frac{T^{\ln\left(1+(1-\frac{1}{m})c\gamma\right)}}{m} \lesssim \epsilon_{\text{arg}} \lesssim \frac{T^{(1-\frac{1}{m})c\gamma}}{m}.$$

3. Above results hold for ϵ_{stab} with a few additional assumptions
4. Above lower bounds hold for the IFT-based algorithm based on its fundamental relation to the UD-based algorithm

An example that induces the lower bounds

Example (Constructed HO example)

- The validation loss and training loss are given by:

$$\ell^{\mathrm{val}}(\boldsymbol{\lambda}, \boldsymbol{\theta}; \boldsymbol{z}) = \ell^{\mathrm{tr}}(\boldsymbol{\lambda}, \boldsymbol{\theta}; \boldsymbol{z}) = \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{A}\boldsymbol{\theta} + \boldsymbol{\lambda}^\top \boldsymbol{\theta} - y\boldsymbol{x}^\top \boldsymbol{\theta},$$

  where $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ is symmetric. The eigenvalues of $\boldsymbol{A}$ are $\gamma_1 \leq \cdots \leq \gamma_d$ where $\gamma_1 < 0$ and $|\gamma_1| \geq |\gamma_d|$. Let $\boldsymbol{v}_1$ be a unit eigenvector for γ_1.

- Let S^{val} and $\tilde{S}^{\mathrm{val}}$ be a pair of twin validation sets differing at the i-th example where

$$\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i) = (\boldsymbol{v}_1, 1), \tilde{\boldsymbol{z}}_i = (\tilde{\boldsymbol{x}}_i, \tilde{y}_i) = (-\boldsymbol{v}_1, 1).$$

Construction of the lower bound I

1. Aligned formulation with the upper bound
   - **Observation:** The upper-bounded recursion

$$\mathbb{E}_{\mathcal{A}}[\|\boldsymbol{\lambda}_t - \tilde{\boldsymbol{\lambda}}_t\|] \leq \left[1 + (1 - 1/m)\alpha_t\gamma\right]\mathbb{E}_{\mathcal{A}}[\|\boldsymbol{\lambda}_{t-1} - \tilde{\boldsymbol{\lambda}}_{t-1}\|] + \frac{2\alpha_t L}{m}$$

   - **Inspiration on the construction:** We need to determine conditions for the hyperparameter divergence exhibiting lower-bounded recursion with an aligned formulation (▶ *lower-bounded expansion properties* in Section 4).

2. Inducing instability for the UD-based algorithm
   - **Observation:** Concavity leads to instability for single-level SGD
   - **Inspiration on the construction:** The compound validation loss $\mathcal{L}$ needs to exhibit concavity in at least one dimension (▶ an "indefinite" second order term).

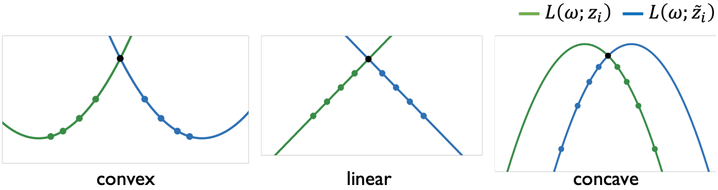

Figure 2.1: Stability of SGD on functions with different convexity

3. Simple bilevel structure for calculating the hyperparameter divergence
    - **Observation:** Bilevel optimization process results in complicated hyperparameter updates (e.g., in the classical ridge regression).
    - **Inspiration on the construction:** The interaction of $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$ needs to be simple (▶ a bilinear cross term).

Example G.1 (Regularization coefficient in ridge regression)**.** The validation loss and training loss are given by $\ell^{\text{val}}(\lambda, \boldsymbol{\theta}) = \frac{1}{2}(y - \boldsymbol{\theta}^T \boldsymbol{x})^2$, $\ell^{\text{tr}}(\lambda, \boldsymbol{\theta}) = \frac{1}{2}(y - \boldsymbol{\theta}^\top \boldsymbol{x})^2 + \frac{\lambda}{2}\boldsymbol{\theta}^\top \boldsymbol{\theta}$. Solving it with UD-based Algorithm 1, we have the inner output as $\boldsymbol{\theta}_K(\lambda) = \prod_{k=1}^{K}(\boldsymbol{I} - \eta\lambda\boldsymbol{I} - \eta\boldsymbol{x}_{j_k}\boldsymbol{x}_{j_k}{}^\top)\boldsymbol{\theta}_0 + \sum_{i=1}^{K}\prod_{l=k+1}^{K}(\boldsymbol{I} - \eta\lambda\boldsymbol{I} - \eta\boldsymbol{x}_{j_l}\boldsymbol{x}_{j_l}{}^\top)\eta y_{j_k}\boldsymbol{x}_{j_k}$ and a far more complex inner Jacobian $\nabla_\lambda\boldsymbol{\theta}_K(\lambda)$, resulting in a unmeasurable hypergradient $\nabla\mathcal{L}(\lambda) = \nabla_\lambda\boldsymbol{\theta}_K(\lambda)(y - \boldsymbol{\theta}_K(\lambda)^\top\boldsymbol{x})(-\boldsymbol{x})$.

Figure 2.2: An example of HO in ridge regrassion

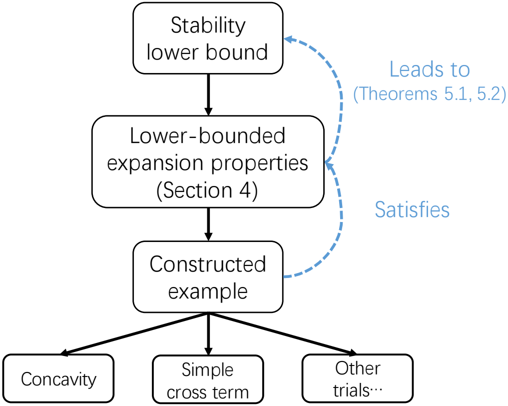

Figure 2.3: Overview of the construction

Thank you for your attention!

Email: wangrz@ruc.edu.cn

Reference

[1] Franceschi L, Frasconi P, Salzo S, et al. Bilevel programming for hyperparameter optimization and meta-learning. ICML, 2018.
[2] Liu H, Simonyan K, Yang Y. DARTS: differentiable architecture search. ICLR, 2019.
[3] Lorraine J, Vicol P, Duvenaud D. Optimizing millions of hyperparameters by implicit differentiation. AISTATS, 2020.
[4] Bao F, Wu G, Li C, et al. Stability and generalization of bilevel programming in hyperparameter optimization. NeurIPS, 2021.