ECE 8101: Nonconvex Optimization for Machine Learning

Lecture Note 5-1: Bilevel Optimization

Jia (Kevin) Liu

Associate Professor
Department of Electrical and Computer Engineering
The Ohio State University, Columbus, OH, USA

Autumn 2024

Outline

In this lecture:

- Motivation and Bilevel Optimization Formulations
- Representative Algorithms
- Convergence Results

Bilevel Optimization (BLO) Formulation

$$\min_{\mathbf{x}\in\mathcal{U}} f\left(\mathbf{x}, \mathbf{y}^*(\mathbf{x})\right)$$
$$\text{s.t.} \quad \mathbf{y}^*(\mathbf{x}) \in \underset{h(\mathbf{x},\mathbf{y})\leq 0}{\operatorname{argmin}}\, g\left(\mathbf{x}, \mathbf{y}\right).$$

- f, g, and h are bivariate *smooth* functions
- $\mathbf{x} \in \mathbb{R}^m$ is the upper-level (UL) variables subject to UL constraint set $\mathcal{U}$
- $\mathbf{y} \in \mathbb{R}^n$ is the lower-level (LL) variables subject to LL constraint $h(\mathbf{x}, \mathbf{y}) \leq 0$
- $\mathbf{y}^*(\mathbf{x})$ is a LL optimal solution

Motivating Examples of BLO

- Meta Learning "learning to learn"
- Hyper-parameter Optimization UL: pick/adjust hyper-para.
  LL: train some learner w/ chosen hp to verify performance
- Actor-critic in RL
  LL Critic: evaluate how good the current policy is (PE)
  UL Actor: improve policy para. (PI)
- Continual Learning
  To avoid catastrophic forgetting (CF): 1. data reply - $\alpha \in [0,1]$ "old data percentage"
  2. regularization $+ \gamma \|\underline{w}\|_2^2$
  3. orthogonal proj.
- Coreset Selection
- Pretraining-Finetuning Pipeline for LLMs and LFMs

BLO Example: Coreset Selection for Model Training

- Goal: Dataset reduction to ease the pain of data storage by identifying the most informative subset of data
- Consists of two tasks:
  - (T1): Select the most representative data samples to form coreset
  - (T2): Validating the performance of the selected coreset in model training
- Problem formulation:

$$\min_{\mathbf{w}\in\mathcal{U}} \ell_{\text{val}}\left(\boldsymbol{\theta}^*(\mathbf{w})\right)$$
$$\text{s.t.} \quad \boldsymbol{\theta}^*(\mathbf{w}) \in \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, l_{\text{tr}}\left(\boldsymbol{\theta}, \mathbf{w}\right).$$

  - $\mathbf{w}$ represents weight vector for data selection, with $w_i = 0$ meaning the i-th sample is not selected
  - $\boldsymbol{\theta}$ represents model training parameters

Challenge: The UL and LL tasks are intertwined!

BLO Can Be Intractable

- A non-convex BLO problem even if UL and LL problems are convex:

$$\min_{x\in[-1,1]} \; x^2 - xy^*(x) - y^*(x)^2$$
$$\text{s.t.} \quad y^*(x) = \operatorname*{argmin}_{y\in[-1,1], x-y=0} \; -(x^2 - xy - y^2).$$

  - Both UL and LL problems are strongly convex
  - It's trivial to see that $y^*(x) = x$
  - As a result, the UL objective function can be expressed as $\ell(x) = -x^2$, a non-convex optimization (reverse convex)
  - Source of difficulty: The coupling constraint $x - y = 0$

- If the $x - y = 0$ constraint is removed, the problem becomes classical min-max (saddle-point) problem:

$$\min_{x\in[-1,1]} \max_{y\in[-1,1]} \; x^2 - xy - y^2$$

Only focus on the subset of tractable BLO problems in this course

Classes of Tractable BLO with Special Structures

1. The LL constraint set, if present, is linear and only related to $\mathbf{y}$:

$$h(\mathbf{x}, \mathbf{y}) = \mathbf{A}\mathbf{y} - \mathbf{b}$$

for some matrix $\mathbf{A}$ and vector $\mathbf{b}$ of conformal dimensions

2. The solution of the LL problem is a singleton
   - Often assume an even stronger condition that the LL objective $g(\cdot, \cdot)$ is strongly convex in $\mathbf{y}$
   - Relaxation to strictly convex settings is still being actively researched

Two Major Classes of BLO Problems in ML Literature

- LL-Unconstrained BLO (LU-BLO)

$$\begin{aligned} &\min_{\mathbf{x}\in\mathcal{U}} f\left(\mathbf{x}, \mathbf{y}^*(\mathbf{x})\right) \\ &\text{s.t.} \quad \mathbf{y}^*(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{y}\in\mathbb{R}^n} g\left(\mathbf{x}, \mathbf{y}\right). \end{aligned}$$

- LL-Constrained BLO (LC-BLO)

$$\begin{aligned} &\min_{\mathbf{x}\in\mathcal{U}} f\left(\mathbf{x}, \mathbf{y}^*(\mathbf{x})\right) \\ &\text{s.t.} \quad \mathbf{y}^*(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{y}\in\mathcal{C}} g\left(\mathbf{x}, \mathbf{y}\right), \end{aligned}$$

where $\mathcal{C} := \{\mathbf{y} | \mathbf{A}y - \mathbf{b} \leq \mathbf{0}\}$.

LC-BLO could be much harder than LU-BLO

Connection of BLO with Game Theory

- BLO has strong ties with Stackelberg (or leader-following) games
  - Two players: leader and follower
  - Leader acts first to maximize its utility based on its knowledge of follower's anticipated response
  - Follower acts second to maximize its utility based on leader's action
  - Identifying a solution (i.e., Stackelberg equilibrium) can be cast as BLO
  - BLO also admits a Stackelberg game-theoretic interpretation (UL and LL problems correspond to identifying optimal leader and follower actions, respectively)

- Special case of Stackelberg game: min-max optimization

$$\min_{\mathbf{x}\in\mathcal{U}} \max_{\mathbf{y}\in\mathcal{C}} f(\mathbf{x}, \mathbf{y})$$

  - Also referred to as saddle point problem
  - Min-max is also a special case of BLO with $g = -f$
  - Also highly relevant and extensively studied in the ML literature

Implicit Gradient (IG)

- Reasons to consider LU-BLO and LC-BLO: UL objectives of both problems are (potentially) differentiable w.r.t. $\mathbf{y}$
- Suppose Jacobian $\frac{d\mathbf{y}^*(\mathbf{x})}{d\mathbf{x}}$ exists, by chain rule:

$$\frac{df(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))}{d\mathbf{x}} = \nabla_{\mathbf{x}} f(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) + \underbrace{\frac{d\mathbf{y}^*(\mathbf{x})}{d\mathbf{x}}^\top}_{\text{IG}} \nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))$$

"hyper-gradient"

  - IG characterizes gradient of argmin-based LL objective w.r.t. UL variable $\mathbf{x}$
  - Note: IG does not always exist (even for LU-BLO and LC-BLO)
  - Stronger assumptions are need for IG to exist (e.g., $g(\cdot, \cdot)$ is strongly convex)
- Even if IG exists, its computation is very different in LU-BLO and LC-BLO
  - LU-BLO: IG can be expressed in closed-form using Implicit Function Theorem
  - LC-BLO: IG has no closed-form in general
- For min-max problems with unconstrained LL problem: IG can be ignored
  - $\nabla_{\mathbf{y}} f(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) = \mathbf{0}$ since $\nabla_{\mathbf{y}} g(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) = \mathbf{0}$ and $g = -f$

BLO with Non-Singleton LL Solutions (NS-BLO)

- NS-BLO can be written as:

$$\min_{\mathbf{x}\in\mathcal{U},\mathbf{y}'\in\mathcal{S}(\mathbf{x})} f\left(\mathbf{x},\mathbf{y}'\right)$$
$$\text{s.t.} \quad \mathcal{S}(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{y}\in\mathcal{C}} g\left(\mathbf{x},\mathbf{y}\right)$$

  - $\mathcal{S}(\mathbf{x})$ denotes LL solution set
  - Much harder b/c optimization over $\mathbf{y}$ is coupled across UL and LL objectives

Three Main Approaches for Solving LU-BLO and LC-BLO

- The Implicit Function (IF)-Based Approach
  - Use Implicit Function Theorem to calculate IG
- The Gradient Unrolling (GU)-Based Approach
  - Unrolling a given algorithm with a fixed number of steps to approximate IG
- The Value-Function (VF)-Based Approach
  - Reformulate BLO as a single-level regularized optimization problem

IF-Based Approach for LU-BLO

- Consider LU-BLO problems with singleton LL solutions and the LL problem is strongly convex in $\mathbf{y}$ (LLSC)
  - Some applications may have a strongly regularized function (e.g., $\gamma\|\mathbf{y}\|_2^2$ with large enough γ so LLSC is satisfied

- The IG can be computed as

$$\frac{d\mathbf{y}^*(\mathbf{x})^\top}{d\mathbf{x}} = -\nabla^2_{\mathbf{x},\mathbf{y}} g(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) \nabla^2_{\mathbf{y},\mathbf{y}} g(\mathbf{x}, \mathbf{y}^*(\mathbf{x}))^{-1}$$

  - IG computation involves Jacobian $\nabla^2_{\mathbf{x},\mathbf{y}} g$ and Hessian inverse $\nabla^2_{\mathbf{y},\mathbf{y}} g$
  - Both could be challenging to compute in practice
  - Different IF-based approaches use different techniques to approximate IG

$y^*(\underline{x})$: $\nabla_y g(\underline{x}, y^*(\underline{x})) = \underline{0}$

taken der. w.r.t. $\underline{x}$ $\Rightarrow$ $\nabla^2_{xy} g(\underline{x}, y^*(\underline{x})) + \nabla^2_{yy} g(\underline{x}, y^*(\underline{x})) \frac{dy^*(\underline{x})}{d\underline{x}} = 0$

Solve for $\frac{dy^*(\underline{x})}{d\underline{x}}$

Basic IF-Based Framework for Solving LU-BLO

In each iteration t:

1. LL Optimization: Given $\mathbf{x}_t$, obtain an approximate LL solution $\hat{\mathbf{y}}(\mathbf{x}_t)$
2. Hyper-gradient Approximation: Based on $\hat{\mathbf{y}}(\mathbf{x}_t)$, compute approximate Jacobian and Hessian inverse: $\hat{\nabla}^2_{\mathbf{x},\mathbf{y}} g(\mathbf{x}_t, \hat{\mathbf{y}}(\mathbf{x}_t))$ and $\hat{\nabla}^2_{\mathbf{y},\mathbf{y}} g(\mathbf{x}_t, \hat{\mathbf{y}}(\mathbf{x}_t))^{-1}$
3. Compute approximate hyper-gradient as

$$\hat{\nabla} f(\mathbf{x}_t) = \nabla_{\mathbf{x}} f(\mathbf{x}_t, \hat{\mathbf{y}}(\mathbf{x}_t)) - \hat{\nabla}^2_{\mathbf{x},\mathbf{y}} g(\mathbf{x}_t, \hat{\mathbf{y}}(\mathbf{x}_t)) \underbrace{\hat{\nabla}^2_{\mathbf{y},\mathbf{y}} g(\mathbf{x}_t, \hat{\mathbf{y}}(\mathbf{x}_t))^{-1}}_{\mathbf{H}^{-1}} \underbrace{\nabla_{\mathbf{y}} f(\mathbf{x}_t, \hat{\mathbf{y}}(\mathbf{x}_t))}_{\mathbf{g}}$$

4. UL Optimization: Update UL variable: $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha \hat{\nabla} f(\mathbf{x}_t)$

Main computational cost stems from computing $\mathbf{H}^{-1}\mathbf{g}$

Approaches to Approximate $\mathbf{H}^{-1}\mathbf{g}$

- Approach 1: The conjugate gradient (CG) approach
  - Map $\mathbf{H}^{-1}\mathbf{g}$ to the solution of quadratic program (QP) $\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^\top \mathbf{H}\mathbf{x} - \mathbf{g}^\top \mathbf{x}$
  - Use FO methods to numerically solve the QP to approximate $\mathbf{H}^{-1}\mathbf{g}$
  - Convergence speed depends on the smallest eigenvalue of the PSD matrix $\mathbf{H}$

- Approach 2: The Sherman-Morrison-Woodbury approach to compute $\mathbf{H}^{-1}$
  - Suppose $\mathbf{B} = \mathbf{A} + \mathbf{UV}$, where $\mathbf{A}^{-1}$ is known or easily computable, $\mathbf{U}$ and $\mathbf{V}$ are low-rank matrices, and $\mathbf{I} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U}$ is invertible
  - Then $\mathbf{B}^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{I} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{A}^{-1}$
  - Special Case: If rank-one update (i.e., $\mathbf{U}$ and $\mathbf{V}$ become $\mathbf{u}$ and $\mathbf{v}$, respectively), then $\mathbf{B}^{-1} = (\mathbf{A} + \mathbf{u}\mathbf{v}^\top)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^\top\mathbf{A}^{-1}}{1+\mathbf{v}^\top\mathbf{A}^{-1}\mathbf{u}}$

Approaches to Approximate $\mathbf{H}^{-1}\mathbf{g}$

- Approach 3: The Neumann-series approximation approach
  - If $\|\mathbf{H}\| \leq 1$, then $\sum_{i=0}^{K}[\mathbf{I}-\mathbf{H}]^i \to \mathbf{H}^{-1}$ as $K \to \infty$
  - Popular for approximating $\mathbf{H}^{-1}$ (using finite K) in stochastic setting (i.e., the UL and LL objectives are associated with stochastic oracle
  - Choose k uniformly randomly from $\{0,\ldots,K-1\}$ and access batch samples $\{g(\mathbf{x},\mathbf{y};\xi_k)\}_{i=1}^{k}$ and compute:

$$\mathbf{H}^{-1} \approx \frac{k}{L_g}\prod_{i=1}^{k}\left(\mathbf{I}-\nabla^2_{\mathbf{y},\mathbf{y}}g(\mathbf{x},\mathbf{y};\xi_i)/L_g\right)$$

  - Biased estimator of $\mathbf{H}^{-1}$ but bias decreases exponentially with K

- Approach 4: Hessian-free approach
  - If LU-BLO is in the form of

$$\min_{\mathbf{x}\in\mathcal{U}} f\left(\mathbf{x},\mathbf{y}^*(\mathbf{x})\right),\ \text{s.t. } \mathbf{y}^*(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{y}\in\mathbb{R}^n} g\left(\mathbf{x},\mathbf{y}\right) + \frac{\lambda}{2}\|\mathbf{y}\|_2^2$$

  - Assumes $\hat{\nabla}^2_{\mathbf{y},\mathbf{y}}g(\mathbf{x}_t,\hat{\mathbf{y}}(\mathbf{x}_t)) \approx \mathbf{0}$ (reasonable when LL objective g involves deep model, e.g., DNN with ReLU activation, where decision boundary is piece-wise linear in a tropical hyper-surface [Alfarra et al. '22]). Hence, $\mathbf{H} \approx \lambda\mathbf{I}$

Extension of IF-Based Approaches to LC-BLO

- IG no longer has closed-form since $\nabla_{\mathbf{y}} g(\mathbf{x}, \mathbf{y}^*(\mathbf{x})) = \mathbf{0}$ does not hold
- Example: Consider the following LC-BLO:

$$\min_{x \in [0,1]} x + y^*(x), \text{ s.t. } y^*(x) \in \underset{y \in [\frac{1}{2}, 1]}{\operatorname{argmin}} (x - y)^2$$

  - It's easy to show that $y^*(x) = 1/2$ for $x \leq 1/2$, and $y^*(x) = x$ for $x > 1/2$.
  - At point $x = 1/2$, the mapping $y^*(x)$ is continuous but not differentiable, hence the UL function $x + y^*(x)$ is non-differentiable.

- With additional assumptions on the matrix $\mathbf{A}$ in the constraint set $\mathcal{C}$ of LC-BLO, one can apply IFT to the Karuch-Kuhn-Tucker (KKT) condition of the LL problem to calculate IG [Khanduri et al. ICML'23]
  - IF-based approaches are not suitable for handling general nonlinear constraints in LL problem. VF-based approaches are often employed in this case

GU-Based Approach for LU-BLO

Basic Idea:

- Use an unrolled LL optimizer as an intermediate step to connect LL solution with UL optimization process $\hat{y} \rightarrow y^*$
- Then use automatic differentiation (AD) technique to compute gradients w.r.t. UL variable $\mathbf{x}$

The IG computation depends on the LL optimizer and no longer uses implicit function-based expression

Basic GU-Based Framework for Solving LU-BLO

Let $h(\cdot) : \mathcal{U} \times \mathcal{C} \to \mathcal{C}$ denote one step of an LL algorithm. In each iteration t:

1. LL Optimization: Run K-step LL optimization:

$$\mathbf{y}_k = h(\mathbf{x}_t, \mathbf{y}_{k-1}), \quad k = 1, \ldots, K.$$

Define $\hat{\mathbf{y}}(\mathbf{x}_t) := \mathbf{y}_K = h(\mathbf{x}_t, h(\mathbf{x}_t, \cdots, h(\mathbf{x}_t, \mathbf{y}_0)))$;

2. UL Optimization: Leverage AD to compute the approximate hyper-gradient

$$\hat{\nabla} f(\mathbf{x}_t, \hat{\mathbf{y}}(\mathbf{x}_t)) := \frac{df(\mathbf{x}_t, h(\mathbf{x}_t, h(\mathbf{x}_t, \cdots, h(\mathbf{x}_t, \mathbf{y}_0))))}{d\mathbf{x}}$$

and update UL variable: $\mathbf{x}_{t+1} = \mathbf{x}_t - \alpha \hat{\nabla} f(\mathbf{x}_t, \hat{\mathbf{y}}(\mathbf{x}_t))$

Differences between IF-Based and UG-Based Approaches

Consider the case $h(\cdot)$ is gradient mapping: $h(\mathbf{x}_t, \mathbf{y}_{k-1}) = \mathbf{y}_{k-1} - \beta \nabla_{\mathbf{y}} g(\mathbf{x}_t, \mathbf{y}_{k-1})$ with step size $\beta > 0$ and assume $\mathbf{y}_0$ is independent of $\mathbf{x}$.

- $K = 1$ (i.e., a single GD step is performed for LL problem)): The IG can be computed in closed-form as:

MAML

$$\frac{d\hat{\mathbf{y}}(\mathbf{x}_t)}{d\mathbf{x}} = \frac{d[\mathbf{y}_0 - \beta \times \nabla_{\mathbf{y}} g(\mathbf{x}_t, \mathbf{y}_0)]}{d\mathbf{x}} = -\beta \nabla^2_{\mathbf{x},\mathbf{y}} g(\mathbf{x}_t, \mathbf{y}_0)$$

- $K = 2$: The IG can be computed in closed-form as:

$$\frac{d\hat{\mathbf{y}}(\mathbf{x}_t)}{d\mathbf{x}} = \frac{d[\mathbf{y}_1 - \beta \times \nabla_{\mathbf{y}} g(\mathbf{x}_t, \mathbf{y}_1)]}{d\mathbf{x}} = -\beta[\mathbf{I} + \beta \times \nabla^2_{\mathbf{y},\mathbf{y}} g(\mathbf{x}_t, \mathbf{y}_1)] \nabla^2_{\mathbf{x},\mathbf{y}} g(\mathbf{x}_t, \mathbf{y}_0)$$

Hessian inverse is not needed, but computational and memory requirements rapidly increase as the number of unrolling steps increases

Practical Considerations for UG-Based Approaches

When K is too large, manual unrolling become necessary to save memory and computational costs

- Forward Gradient Unrolling (FGU): The Jacobian $\mathbf{y}_K$ w.r.t. $\mathbf{x}$ (IG approx.) is:

$$\underbrace{\frac{d\mathbf{y}_K}{d\mathbf{x}}}_{\mathbf{Z}_K} = \underbrace{\frac{\partial \mathbf{y}_K}{\partial \mathbf{y}_{K-1}}}_{\mathbf{A}_K} \underbrace{\frac{d\mathbf{y}_{K-1}}{d\mathbf{x}}}_{\mathbf{Z}_{K-1}} + \underbrace{\frac{\partial \mathbf{y}_K}{\partial \mathbf{x}}}_{\mathbf{B}_K}$$

  or in iterative form: $\mathbf{Z}_k = \mathbf{A}_k \mathbf{Z}_{k-1} + \mathbf{B}_k$, $k = 1, 2, \ldots, K$, with $\mathbf{Z}_0 = \frac{\mathbf{y}_0}{d\mathbf{x}} = \mathbf{0}$ assuming $\mathbf{y}_0$ is independent of $\mathbf{x}$.

  - Both $\mathbf{A}_k$ and $\mathbf{B}_k$ can be computed along with the k-th LL step $\mathbf{y}_k = h(\mathbf{x}_t, \mathbf{y}_{k-1})$ and discarded immediately after $\mathbf{Z}_k$ is obtained, thus significantly saving memory cost as K gets large
  - However, FGU needs to keep track of $\mathbf{A}_k$, $\mathbf{B}_k$, and $\mathbf{Z}_{k-1}$, which may still be expensive for high-dimensional $\mathbf{x}$ and $\mathbf{y}$ variables

Practical Considerations for UG-Based Approaches

To save computation costs for high-dimensional $\mathbf{x}$ and $\mathbf{y}$:

- Backward Gradient Unrolling (BGU): Instead of computing IG explicitly, BGU directly obtains gradient of UL variable in the following iterative fashion:

$$\frac{df(\mathbf{x},\mathbf{y}_K)}{d\mathbf{x}} = \underbrace{\frac{\partial f(\mathbf{x},\mathbf{y}_K)}{\partial \mathbf{x}}}_{\mathbf{c}_K} + \underbrace{\frac{d\mathbf{y}_K^\top}{d\mathbf{x}}}_{\mathbf{Z}_K^\top} \underbrace{\frac{\partial f(\mathbf{x},\mathbf{y}_K)}{\partial \mathbf{y}_K}}_{\mathbf{d}_K} \overset{\text{(FGU)}}{=} \mathbf{c}_K + \underbrace{(\mathbf{Z}_{K-1}^\top \mathbf{A}_K^\top + \mathbf{B}_K^\top)}_{\mathbf{Z}_K^\top} \mathbf{d}_K$$

$$= \underbrace{(\mathbf{c}_K + \mathbf{B}_K^\top \mathbf{d}_K)}_{\mathbf{c}_{K-1}} + \mathbf{Z}_{K-1}^\top \cdot \underbrace{\mathbf{A}_K^\top \mathbf{d}_K}_{\mathbf{d}_{K-1}} = \mathbf{c}_{K-1} + \mathbf{Z}_{K-1}^\top \mathbf{d}_{K-1}$$

$$= \cdots = \mathbf{c}_0 + \mathbf{Z}_0^\top \mathbf{d}_0 = \mathbf{c}_{-1},$$

  - $\mathbf{c}_{k-1} = \mathbf{c}_k + \mathbf{B}_k^\top \mathbf{d}_k$, $k = 0, 1, \ldots, K$, with $\mathbf{c}_K = \frac{\partial f(\mathbf{x},\mathbf{y}_K)}{\partial \mathbf{x}}$
  - $\mathbf{d}_{k-1} = \mathbf{A}_k^\top \mathbf{d}_k$, $k = 0, 1, \ldots, K$, with $\mathbf{d}_K = \frac{f(\mathbf{x},\mathbf{y}_K)}{\partial \mathbf{y}_K}$

- BGU only requires storing $(\mathbf{c}_k, \mathbf{d}_k)$ vectors throughout the recurssion by using the Jacobian-vector product trick, thus being more advantageous for problems with high-dimensional $\mathbf{x}$ and $\mathbf{y}$ than FGU. However, BGU still needs to store all unrolling steps $\{\mathbf{y}_k\}_{k=1}^K$ and may not be efficient for large K

The Value Function (VF)-Based Approach for BLO

- VF-based methods also do not need to compute Hessian inverse
- Key Idea: Reformulate BLO into constrained single-level optimization

$$\min_{\mathbf{x},\mathbf{y}\in\mathcal{C}} f(\mathbf{x},\mathbf{y}),\ \text{s.t.}\ g(\mathbf{x},\mathbf{y}) \leq g^*(\mathbf{x}),$$

where $g^*(\mathbf{x}) := \min_{\mathbf{y}\in\mathcal{C}} g(\mathbf{x},\mathbf{y})$ is referred to as the value function (VF)

  - Challenge: $g^*(\mathbf{x})$ is not necessarily smooth and can be non-convex

- A relaxed version: replace $g^*(\mathbf{x})$ with a smooth surrogate

$$g^*_{\boldsymbol{\mu}}(\mathbf{x}) = \min_{\mathbf{y}\in\mathcal{C}} g(\mathbf{x},\mathbf{y}) + \frac{\mu_1}{2}\|\mathbf{y}\|_2^2 + \mu_2$$

  - $\boldsymbol{\mu} := \{\mu_1, \mu_2\}$ is a pair of positive constants to induce smoothness of $g^*_{\boldsymbol{\mu}}(\mathbf{x})$
  - With the relaxed VF formulation, one can adopt standard nonlinear optimization algorithms (e.g., penalty-based or interior-point methods)

Convergence Metrics of BLO

- LU-BLO:
  - Focus on the notion of ϵ-stationary of UL hyper-gradient
  - Deterministic setting: A UL solution $\bar{\mathbf{x}}$ is ϵ-stationary if $\|\nabla f(\bar{\mathbf{x}}, \mathbf{y}^*(\bar{\mathbf{x}}))\|_2^2 \leq \epsilon$
  - Stochastic setting: A UL solution $\bar{\mathbf{x}}$ is ϵ-stationary if $\mathbb{E}[\|\nabla f(\bar{\mathbf{x}}, \mathbf{y}^*(\bar{\mathbf{x}}))\|_2^2] \leq \epsilon$, where $\mathbb{E}[\cdot]$ is taken overall all randomness of the algorithm
  - Note: When UL problem in LU-BLO is constrained (i.e., $\mathcal{U} \subset \mathbb{R}^m$), then the UL objective $f(\bar{\mathbf{x}}, \mathbf{y}^*(\bar{\mathbf{x}}))$ may not be differentiable over $\bar{\mathbf{x}}$ in general

- LC-BLO:
  - If using the IF-based approach and if IF is differentiable, similar ϵ-stationarity can be used
  - If IF is non-differentiable, can use subgradient optimality, proximal gradient methods, and Moreau envlope techniques
  - If using VF-based approaches, a widely used stationarity metric is the KKT stationarity

- Further, one often considers oracle complexity to quantify the number of gradient evaluations to achieve ϵ-stationarity

Convergence Results of Methods for Solving LU-BLO

Classification of methods for solving LU-BLO:

- Deterministic vs. Stochastic
  - IF-based methods: Replace UL and LL gradients by appropriate stochastic gradient estimates. However, obtaining unbiased estimator for Hessian inverse in IG computation is challenging
- Single-loop vs. Double-loop
  - Single-loop: Only performs a fixed number of steps for LL updates before every UL update
  - Double-loop: As many LL updates steps as need to obtain a very accurate approximation of $\mathbf{y}^*(\mathbf{x})$
  - Single-loop is easy to implement, while double-loop is easy to analyze
- Vanilla SGD vs. Momentum-based SGD vs. VR-based SGD
  - Momentum-based and VR-based methods typically have better theoretical convergence rate

IF-Based Stochastic Method for LU-BLO & LC-BLO

Given initial $\mathbf{x}_0$ and iteration number T; In each iteration t:

1. LL Optimization: Given $\mathbf{x}_t$, call vanilla SGD, momentum-based SGD, or VR-based SGD to obtain LL solution $\hat{\mathbf{y}}(\mathbf{x}_t)$

2. Approximation: Compute stochastic estimate of UL hypergradient:
   - Get stochastic versions of $\nabla_{\mathbf{x}} f(\mathbf{x}_t, \hat{\mathbf{y}}(\mathbf{x}_t)), \nabla_{\mathbf{y}} f(\mathbf{x}_t, \hat{\mathbf{y}}(\mathbf{x}_t)), \hat{\nabla}^2_{\mathbf{x},\mathbf{y}} g(\mathbf{x}_t, \hat{\mathbf{y}}(\mathbf{x}_t))$
   - Approximate Hessian inverse $\hat{\nabla}^2_{\mathbf{y},\mathbf{y}} g(\mathbf{x}_t, \hat{\mathbf{y}}(\mathbf{x}_t))^{-1}$
   - Obtain stochastic estimate of UL hyper-gradient $\hat{\nabla} f(\mathbf{x}_t, \xi_t)$ for $\mathbf{x}_t$

3. UL Optimization: Call vanilla SGD, momentum-based SGD, or VR-based SGD to update $\mathbf{x}_t$

Convergence Results of BLO Methods

Stochastic BLO

Method	Principle	Loop	UL OC	LL OC
BSA [Ghadimi & Wang '18]	IF	Double	$O(\epsilon^{-2})$	$O(\epsilon^{-3})$
TTSA [Hong et al. '20]	IF	Single	$O(\epsilon^{-2.5})$	$O(\epsilon^{-2.5})$
stocBio [Ji et al. ICML'21]	IF	Double	$O(\epsilon^{-2})$	$O(\epsilon^{-2})$
SOBA [Dagreou et al. NeurIPS'22]	IF	Single	$O(\epsilon^{-2})$	$O(\epsilon^{-2})$
ALSET [Chen et al. 21]	IF	Single	$O(\epsilon^{-2})$	$O(\epsilon^{-2})$
F^{2}SA [Kwon et al. 23]	VF	Single	$O(\epsilon^{-3.5})$	$O(\epsilon^{-3.5})$
AmIGO [Arbel & Mairal, ICLR'22]	IF	Double	$O(\epsilon^{-2})$	$O(\epsilon^{-2})$

Momentum-Based and VR-Based BLO

Method	Principle	Loop	UL OC	LL OC
STABLE [Chen et al. AISTATS'22]	IF	Single	$O(\epsilon^{-2})$	$O(\epsilon^{-2})$
SUSTAIN [Khanduri et al. NeurIPS'21]	IF	Single	$O(\epsilon^{-1.5})$	$O(\epsilon^{-1.5})$
VRBO [Yang et al. NeurIPS '21]	IF	Double	$O(\epsilon^{-1.5})$	$O(\epsilon^{-1.5})$
SABA [Dagreou et al. NeurIPS'22]	IF	Double	$O(N^{2/3}\epsilon^{-1})$	$O(N^{2/3}\epsilon^{-1})$
F^{3}SA [Kwon et al. 23]	VF	Single	$O(\epsilon^{-2.5})$	$O(\epsilon^{-2.5})$
SBFW [Akhtar et al. '21]	IF	Single	$O(\epsilon^{-4})$	$O(\epsilon^{-4})$

Convergence Results of BLO Methods

Deterministic BLO

Method	Principle	Loop	UL OC	LL OC
BA [Ghadimi & Wang '18]	IF	Single	$O(\epsilon^{-1})$	$O(\epsilon^{-1.25})$
AID-BIO [Ji et al. ICML'21]	IF	Single	$O(\epsilon^{-1})$	$O(\epsilon^{-1})$
ITD-BIO [Ji et al. ICML'21]	GU	Double	$O(\epsilon^{-1})$	$O(\epsilon^{-1})$
MSTSA [Khanduri et al. '21]	IF	Single	$O(\epsilon^{-1})$	$O(\epsilon^{-1})$
K-RMD [Shaban et al. AISTATS'19]	GU	Double	$O(\epsilon^{-2})$	$O(K\epsilon^{-2})$
FGU/BGU [Franceschi et al. ICML'21]	GU	Double	N/A	N/A