Fully First-Order Methods for Bilevel Optimization

Jeongyeol Kwon, Dohyun Kwon, Stephen Wright, Rob Nowak, Hanbaek Lyu

(UW-Madison)

TTIC, Halloween, 2025

Sources

- J. Kwon, D. Kwon, S. Wright, R. Nowak, *A Fully First-Order Methods for Stochastic Bilevel Optimization*, ICML 2023 (Oral)
- J. Kwon, D. Kwon, S. Wright, R. Nowak, *On Penalty-Methods for Nonconvex Bilevel Optimization and First-Order Stochastic Approximation*, ICLR 2024 (Spotlight)
- J. Kwon, D. Kwon, H. Lyu, *On the Complexity of First-Order Methods in Stochastic Bilevel Optimization*, ICML 2024
- B. Liu, M. Ye, S. Wright, P. Stone, and Q. Liu, 2022. *BOME! Bilevel optimization made easy: A simple first-order approach*, NeurIPS 2022

The Problem

$$\min_{x,y} f(x,y) \quad \text{subject to} \quad y \in \arg\min_{z} g(x,z).$$

- $\min_{x,y} f(x,y)$ is the "upper-level problem" (Leader)
- $\min_{z} g(x,z)$ is the "lower-level problem" (Follower)

Some examples follow.

Outline

I Examples, Motivation, Approach

II Case of $g(x, \cdot)$ uniformly strongly convex

- Approach
- Algorithms
- Convergence and Complexity
- Lower Bounds
- Numerical Example

III $g(x, \cdot)$ nonconvex (but other conditions hold).

Ex) Power Grid Attack and Defense

Version 1:

- Leader (Defender) allocates resources x to minimize severity of attack (measured by f)
- Follower (Attacker) finds optimal attack y (measured by g), given defender's choice of resources.

Version 2:

- Leader (Attacker) seeks most destructive attack x with severity+cost of attack measured by f. Possibly constraints on strength of the attack x.
- Follower (Defender) aims to thwart any given attack x using strategy y, minimizing severity measured by g.

Attack on power grid [Kim et al., 2016].

- Attacker increases impedance on selected lines (with limit on power of attack)
- Grid operator sheds load on certain buses (minimally) to restore grid feasibility.
- Severity of attack measured by total load shed.

Attacking a Power Grid

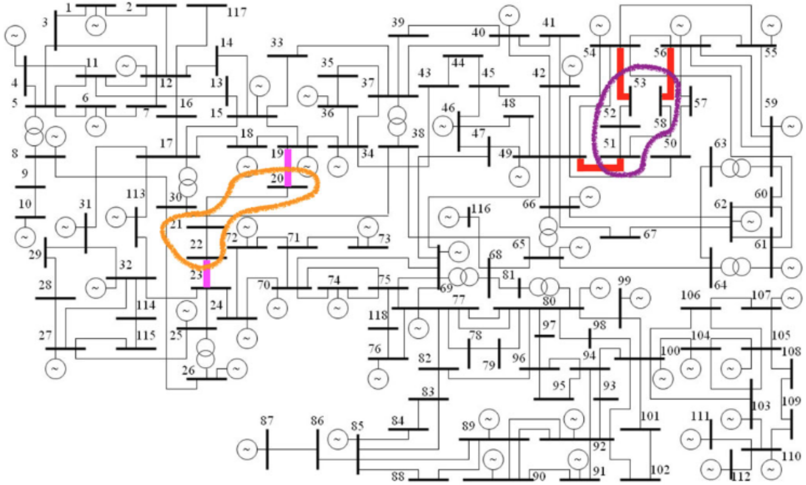
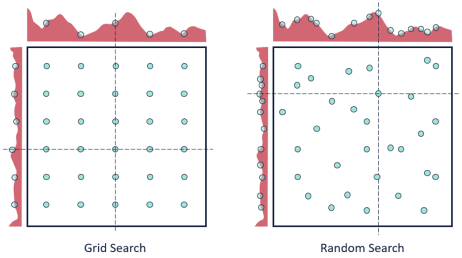


Fig. 5. Voltage disturbance model: attacks on the 118-Bus system. The transmission lines indicated by thicker dark lines are are the optimal attack for $\kappa = 3$. The thick lighter lines are added to the optimal three-line attack when κ is increased to 5. (Image source: [26])

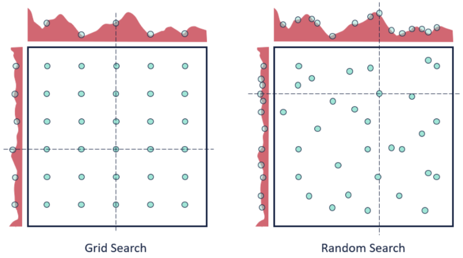
Ex) Hyperparameter Tuning as Bilevel Optimization

Few hyperparameters (e.g. weights on a few different regularizers in the objective function)



Ex) Hyperparameter Tuning as Bilevel Optimization

Few hyperparameters (e.g. weights on a few different regularizers in the objective function)



Many hyperparameters e.g. weight each item of training data according to its reliability.

Bilevel formulation of the latter (see example later). λ contains weights for training data items.

$$\begin{array}{rl} \text{(Validating on trusted data)} & \text{find } \lambda \text{ that minimizes } l_{\mathtt{val}}(\theta^*(\lambda); \lambda) \\ \text{(get } \theta \text{ from training with weights } \lambda\text{)} & \text{where} \quad \theta^*(\lambda) \text{ minimizes } l_{\mathtt{tr}}(\theta; \lambda) \end{array}$$

Ex) Neural Architecture Search (NAS)

Automate design of the neural network structure (e.g., number of layers, connection types).

Outer Problem (Leader): Find the best architecture a from a search space $\mathcal{A}$ that minimizes validation loss.

$$\min_{a \in \mathcal{A}} \mathcal{L}_{\text{val}}(w^*(a))$$

Inner Problem (Follower): For a *given* architecture a, train its weights w^* to convergence on the training data.

$$w^*(a) = \arg\min_{w} \mathcal{L}_{\text{train}}(w, a)$$

Ex) Adversarial Learning & Robustness

Training a model to be robust against "worst-case" attacks. This is a **min-max** problem, a special form of bilevel optimization.

Outer Problem (Leader - "Min"): Find model weights w that minimize the loss, even against the worst-case attack.

$$\min_{w} \mathbb{E}_{(x,y)} \left[\mathcal{L}(f_w(x + \delta^*(w, x, y)), y)\right]$$

Inner Problem (Follower - "Max"): For the *current* model w and input x, find the perturbation δ (within a budget Δ) that *maximizes* the loss.

$$\delta^*(w, x, y) = \arg\max_{\delta \in \Delta} \mathcal{L}(f_w(x + \delta), y)$$

Ex) Inverse Reinforcement Learning (IRL)

Goal: Infer the unknown reward function an expert agent was optimizing, given only their observed trajectories.

Outer Problem (Leader): Find reward function parameters θ such that the resulting optimal policy $\pi^*(\theta)$ matches the expert's behavior π_E.

$$\min_{\theta} \text{Distance}(\pi^*(\theta), \pi_E)$$

(e.g., matching feature expectations, minimizing divergence)

Inner Problem (Follower): For a *given* reward function R_θ, find the optimal policy π^* that maximizes the expected return.

$$\pi^*(\theta) = \arg\max_{\pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{T} R_\theta(s_t, a_t) \right]$$

(Solve an MDP/RL problem with reward R_θ)

Ex) Reward Design & Intrinsic Motivation

Learn an auxiliary or intrinsic reward function to help an agent learn faster or solve sparse-reward tasks.

Outer Problem (Leader): Find intrinsic reward parameters θ that maximize the agent's performance on the **true (extrinsic) environment reward**:

$$\max_{\theta} J_{\text{extrinsic}}(\pi^*(\theta))$$

Inner Problem (Follower): For a *given* intrinsic reward $R_{\text{int}}(\theta)$, train a policy π^* to maximize the **combined reward** (extrinsic + intrinsic):

$$\pi^*(\theta) = \arg\max_{\pi} \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T}(R_{\text{extrinsic}}(s_t, a_t) + R_{\text{int}}(s_t, a_t; \theta))\right]$$

(Solve an RL problem with the combined reward)

Goals

Develop techniques for **bilevel optimization** that are

1. Scalable for large-scale models
2. Can tolerate stochastic (sampling) noise (in gradient estimates, for example)
3. Are potentially extendible to nonconvex problems
4. Have good convergence guarantees and complexity properties.

These goals motivate our focus on first-order methods that depend only on first derivatives $\nabla_x f$, $\nabla_y f$, $\nabla_x g$, $\nabla_y g$ — or unbiased estimates of these quantities.

II: Strongly Convex Case: Setup

$$\min_{x\in\mathbb{R}^{d_x}} F(x) := f(x, y^*(x)) \quad \text{s.t.} \quad y^*(x) \in \arg\min_{y\in\mathbb{R}^{d_y}} g(x,y), \qquad \textbf{(P)}$$

where f and g are continuously differentiable functions with $g(x,y)$ strongly convex in y with modulus $\mu_g > 0$ for all x.

Can characterize $y^*(x)$ explicitly by $\nabla_y g(x, y^*(x)) = 0$. Can substitute $y^*(x)$ explicitly into $f(x,y)$ and do unconstrained minimization of "hyperobjective" $F(x)$. Note that

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

Thus, a (naive) gradient method applied to F will need

- accurate computation of $y^*(x)$ for each iterate x;
- knowledge of second-derivative matrices $\nabla^2_{xy} g$, $\nabla^2_{yy} g$.

Our methods aim to avoid both these requirements.

Literature

Solutions based on direct estimation of $\nabla^2_{xy} g$, $(\nabla^2_{yy} g)^{-1}$
(Wang and Ghadimi, 2018), (Ji et al., 2021), (Chen et al., 2021), (Dagreou et al., 2022), ...

+ Intuitive, good convergence guarantees
− expensive, less flexible in practice

Literature

Solutions based on direct estimation of $\nabla^2_{xy} g$, $(\nabla^2_{yy} g)^{-1}$
(Wang and Ghadimi, 2018), (Ji et al., 2021), (Chen et al., 2021), (Dagreou et al., 2022), ...

- $+$ Intuitive, good convergence guarantees
- $-$ expensive, less flexible in practice

Solutions that avoid direct second-order estimation (*i.e.,* penalty formulation)
(White et al. 1993), (Lu and Mei 2023), (Shen and Chen 2023), ...

- $+$ Faster, easier to implement
- $+$ More flexible to non-convexity / constraints
- $-$ Can attain approximate solutions (exact is harder)

We use the latter formulation.

We start with the case of $g(x, \cdot)$ uniformly strongly convex.

Assumptions: Case of $g(x, \cdot)$ strongly convex

f and g are smooth (e.g. twice Lipschitz cts diff).

- f is $l_{f,1}$-smooth, g is $l_{g,1}$-smooth; F is $l_{F,1}$-smooth
- $\|\nabla_y f(x,y)\| \le l_{f,0}$, $\|\nabla_x f(x,y)\| \le l_{f,0}$, $\|\nabla_x g(x,y)\| \le l_{g,0}$ for all x, y
- $\nabla^2 f$ is $l_{f,2}$-Lipschitz in (x,y), $\nabla^2 g$ is $l_{g,2}$-Lipschitz in (x,y)
- $g(x,y)$ is convex in y with modulus of convexity $\mu_g > 0$ for all x.
- Condition number of $\nabla^2_{yy} g$: $\kappa := l_{g,1}/\mu_g$

For the stochastic setting, we assume that ∇f and ∇g are accessed by estimators $\nabla f(x,y;\xi)$ and $\nabla g(x,y;\phi)$ (for random variables ξ and ϕ) that are unbiased

$$\mathbb{E}_\xi \nabla f(x,y;\xi) = \nabla f(x,y), \quad \mathbb{E}_\phi \nabla g(x,y;\phi) = \nabla g(x,y),$$

with bounded variances:

$$\mathbb{E}_\xi[\|\nabla f(x,y;\xi) - \nabla f(x,y)\|^2] \le \sigma_f^2, \quad \mathbb{E}_\phi[\|\nabla g(x,y;\phi) - \nabla g(x,y)\|^2] \le \sigma_g^2.$$

Penalty Framework

Goal: Find a point x with $\|\nabla F(x)\| \leq \epsilon$. Estimate the number of iterations in terms of ϵ.

Reformulate (**P**) as

$$\min_{x\in\mathbb{R}^{d_x},\, y\in\mathbb{R}^{d_y}} \quad f(x,y) \quad \text{s.t.} \quad g(x,y) - g^*(x) \leq 0, \qquad \textbf{(P')}$$

where $g^*(x) := g(x, y^*(x))$. Work with a penalty function:

$$\mathcal{L}_\lambda(x,y) := f(x,y) + \lambda(g(x,y) - g^*(x)),$$

which is also the Lagrangian for (**P'**) in this case.

But constraint qualifications fail: gradient of the constraint in (**P'**) w.r.t. x, y is zero at the solution.

A Key Result

$$\mathcal{L}_\lambda(x,y) := f(x,y) + \lambda(g(x,y) - g^*(x))$$

inherits the convexity of $g(x,y)$ w.r.t. y for λ sufficiently large. Define

$$\mathcal{L}^*_\lambda(x) := \min_y \mathcal{L}_\lambda(x,y), \quad y^*_\lambda(x) := \arg\min_y \mathcal{L}_\lambda(x,y).$$

Then

$$\begin{aligned}\nabla\mathcal{L}^*_\lambda(x) &= \nabla_x\mathcal{L}_\lambda(x,y^*_\lambda(x)) \\ &= \nabla_x f(x,y^*_\lambda(x)) + \lambda\nabla_x g(x,y^*_\lambda(x)) - \lambda\nabla_x g(x,y^*(x)).\end{aligned} \tag{GRAD}$$

Moreover, for all λ sufficiently large, we have

$$|F(x) - \mathcal{L}^*_\lambda(x)| = O(\kappa/\lambda), \quad \|\nabla F(x) - \nabla\mathcal{L}^*_\lambda(x)\| = O(\kappa^3/\lambda),$$

where $\kappa =$ bound on condition number of $\nabla^2_{yy} g$.

A Key Result

$$\mathcal{L}_\lambda(x, y) := f(x, y) + \lambda(g(x, y) - g^*(x))$$

inherits the convexity of $g(x, y)$ w.r.t. y for λ sufficiently large. Define

$$\mathcal{L}_\lambda^*(x) := \min_y \mathcal{L}_\lambda(x, y), \quad y_\lambda^*(x) := \arg\min_y \mathcal{L}_\lambda(x, y).$$

Then

$$\begin{aligned} \nabla \mathcal{L}_\lambda^*(x) &= \nabla_x \mathcal{L}_\lambda(x, y_\lambda^*(x)) \\ &= \nabla_x f(x, y_\lambda^*(x)) + \lambda \nabla_x g(x, y_\lambda^*(x)) - \lambda \nabla_x g(x, y^*(x)). \end{aligned} \tag{GRAD}$$

Moreover, for all λ sufficiently large, we have

$$|F(x) - \mathcal{L}_\lambda^*(x)| = O(\kappa/\lambda), \quad \|\nabla F(x) - \nabla \mathcal{L}_\lambda^*(x)\| = O(\kappa^3/\lambda),$$

where $\kappa =$ bound on condition number of $\nabla_{yy}^2 g$.

The Idea

To find x with $\|\nabla F(x)\| \leq \epsilon$, look for x with

- $\|\nabla \mathcal{L}_\lambda^*(x)\| \leq \epsilon$ and
- $\lambda = O(1/\epsilon)$.

Approach:

- Approximately minimize $\mathcal{L}_\lambda^*(x)$ for an increasing sequence of λ, using gradient descent;
- maintain approximations to $y_\lambda^*(x)$ and $y^*(x)$, needed in calculation of $\nabla \mathcal{L}_\lambda^*(x)$.

Can view alternatively as finding an approximate solution of the saddle-point problem

$$\min_{x,y} \max_{z} f(x,y) + \lambda(g(x,y) - g(x,z)), \tag{P-SADDLE}$$

for $\lambda = O(1/\epsilon)$.

The Approach

For $\lambda_k \uparrow \infty$, generate a sequence (x_k, y_k, z_k) such that

- $y_k \approx y^*_{\lambda_k}(x_k)$, $z_k \approx y^*(x_k)$
- $x_{k+1} \leftarrow x_k - \xi \alpha_k \nabla \mathcal{L}^*_{\lambda_k}(x_k)$ — where the gradient is replaced by an approximation calculated as in (GRAD) with

$$y^*_{\lambda_k}(x_k) \leftarrow y_k, \quad y^*(x_k) \leftarrow z_k.$$

Main issues: Choosing the penalty parameter sequence $\{\lambda_k\}$ and steplengths α_k (and γ_k for updating z_k) appropriately.

Can get a stochastic-gradient variant by replacing $\nabla_x f$, $\nabla_y f$, $\nabla_x g$, $\nabla_y g$ by unbiased oracles with bounded variance.

Can define a "momentum" variant with accelerated convergence.

F^{2}SA: the most basic variant

- Grad descent steps for z, y alternate with grad descent steps for x.
- λ incremented at each iteration.

Input: step sizes $\{\alpha_k, \gamma_k\}$; multiplier increment $\{\delta_k\}$; inner-loop iteration count T; step-size ratio ξ; initializations λ_0, x_0, y_0, z_0.

```
1: for $k = 0 \ldots K-1$ do
2:     $z_{k+1} \leftarrow z_k - \gamma_k \nabla_y g(x_k, z_k)$
3:     $y_{k+1} \leftarrow y_k - \alpha_k [\nabla_y f(x_k, y_k) + \lambda_k \nabla_y g(x_k, y_k)]$
4:     $x_{k+1} \leftarrow x_k - \xi \alpha_k [\nabla_x f(x_k, y_{k+1}) + \lambda_k (\nabla_x g(x_k, y_{k+1}) - \nabla_x g(x_k, z_{k+1}))]$
5:     $\lambda_{k+1} \leftarrow \lambda_k + \delta_k$
6: end for
```

ξ is the constant ratio between steplengths for the y and x updates.

F^{2}SA: inner loop for y, z

- Perform T inner-loop iterations to improve accuracy of y and z.
- Update x and λ as before.

Input: step sizes $\{\alpha_k, \gamma_k\}$; multiplier increment $\{\delta_k\}$; inner-loop iteration count T; step-size ratio ξ; initializations λ_0, x_0, y_0, z_0

```
1: for $k = 0 \ldots K-1$ do
2:     $z_{k,0} \leftarrow z_k$, $y_{k,0} \leftarrow y_k$
3:     for $t = 0 \ldots T-1$ do
4:         $z_{k,t+1} \leftarrow z_{k,t} - \gamma_k \nabla_y g(x_k, z_{k,t})$
5:         $y_{k,t+1} \leftarrow y_{k,t} - \alpha_k [\nabla_y f(x_k, y_{k,t}) + \lambda_k \nabla_y g(x_k, y_{k,t})]$
6:     end for
7:     $z_{k+1} \leftarrow z_{k,T}$, $y_{k+1} \leftarrow y_{k,T}$
8:     $x_{k+1} \leftarrow x_k - \xi \alpha_k [\nabla_x f(x_k, y_{k+1}) + \lambda_k (\nabla_x g(x_k, y_{k+1}) - \nabla_x g(x_k, z_{k+1}))]$
9:     $\lambda_{k+1} \leftarrow \lambda_k + \delta_k$
10: end for
```

F^{2}SA: replace gradients by unbiased estimates

Exact gradients not available. e.g. replace $\nabla_x f(x_k, y_{k+1})$ by unbiased estimate $h_{fx}^k := \nabla_x f(x_k, y_{k+1}; \xi_{fx}^k)$ for some random variable ξ_{fx}^k.

Input: step sizes $\{\alpha_k, \gamma_k\}$; multiplier increment $\{\delta_k\}$; inner-loop iteration count T; step-size ratio ξ; initializations λ_0, x_0, y_0, z_0

```
1: for $k = 0 \ldots K-1$ do
2:     $z_{k,0} \leftarrow z_k$, $y_{k,0} \leftarrow y_k$
3:     for $t = 0 \ldots T-1$ do
4:         $z_{k,t+1} \leftarrow z_{k,t} - \gamma_k h_{gz}^{k,t}$
5:         $y_{k,t+1} \leftarrow y_{k,t} - \alpha_k (h_{fy}^{k,t} + \lambda_k h_{gy}^{k,t})$
6:     end for
7:     $z_{k+1} \leftarrow z_{k,T}$, $y_{k+1} \leftarrow y_{k,T}$
8:     $x_{k+1} \leftarrow x_k - \xi \alpha_k (h_{fx}^k + \lambda_k (h_{gxy}^k - h_{gxz}^k))$
9:     $\lambda_{k+1} \leftarrow \lambda_k + \delta_k$
10: end for
```

Analysis

Analysis is based on showing decrease with k in the Lyapunov function

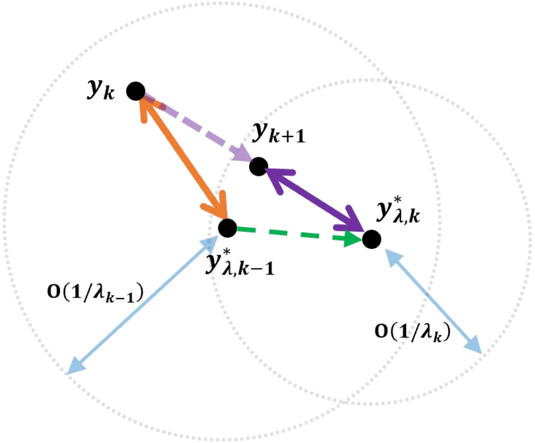
$$\mathbb{V}_k := (F(x_k) - F^*) + \lambda_k l_{g,1} \|y_k - y^*_{\lambda_k}(x_k)\|^2 + \frac{1}{2}\lambda_k l_{g,1} \|z_k - y^*(x_k)\|^2,$$

where F^* is the minimum value of F.

(For the stochastic versions, use expectations.)

$\mathbb{V}_k$ shows the importance of shrinking $\|y_k - y^*_{\lambda_k}(x_k)\|$ and $\|z_k - y^*(x_k)\|$ as $k \to \infty$.

Main Theorem: Gives upper bounds on parameters α_k, γ_k, δ_k, and ξ/T, in terms of problem-dependent constants (and each other).


y_k
y_{k+1}
$y^*_{\lambda,k}$
$y^*_{\lambda,k-1}$
$O(1/\lambda_{k-1})$
$O(1/\lambda_k)$

Ultimately, choose

$$T = O(\kappa), \quad \xi = 1, \quad \alpha_k \sim (k + k_0)^{-a}, \quad \gamma_k \sim (k + k_0)^{-c},$$

where $0 \leq c \leq a \leq 1$ and k_0 is an instance-dependent constant. Also, δ_k is chosen so that

$$\lambda_k = O((k + k_0)^{a-c})$$

(Recall: κ is bound on condition number of $\nabla^2_{yy} g(x, \cdot)$.)

The best choices of a and c depend on the setting.

Complexity of F^{2}SA

Suppose F^{2}SA is run for K iterations and R is drawn randomly from $\{0, 1, \ldots, K\}$. Then

- If gradients of both f and g contain stochastic noise, then set $a = 5/7$ and $c = 4/7$ to get $\mathbb{E}\|\nabla F(x_R)\| \leq \tilde{O}(K^{-1/7})$.
- If gradient of f contains stochastic noise but gradients of g are exact, then set $a = 3/5$ and $c = 2/5$ to get $\mathbb{E}\|\nabla F(x_R)\| \leq \tilde{O}(K^{-1/5})$.
- If gradients of both f and g are exact, set $a = 1/3$ and $c = 0$ to get $\mathbb{E}\|\nabla F(x_R)\| \leq \tilde{O}(K^{-1/3})$.

Number of iterations to get $\mathbb{E}\|\nabla F(x_R)\| \leq \epsilon$ is ϵ^{-7}, ϵ^{-5}, and ϵ^{-3}, respectively.

The $\tilde{O}(\epsilon^{-3})$ result for deterministic gradients contrasts with $O(\epsilon^{-2})$ to find an ϵ-approximate stationary point for smooth nonconvex minimization.

Closing the Gap: [Chen et al., 2025] get $\tilde{O}(\epsilon^{-2})$ for a slightly modified version of F^{2}SA.

[Chen et al., 2025]: $\tilde{O}(\epsilon^{-2})$

Key observation is that $\nabla^2 \mathcal{L}_\lambda(x)$ is $O(\kappa^3)$ — no dependence on λ, for λ sufficiently large.

Follows from

$$\|y^*(x) - y^*_\lambda(x)\| = O(1/\lambda), \quad \|\nabla y^*(x) - \nabla y^*_\lambda(x)\| = O(1/\lambda)$$

Thus, can take longer steps in x than in y or z. Their algorithm looks like F^{2}SA with the inner loop, except:

- λ is fixed at $O(1/\epsilon)$;
- Steps in x is (constantly) $O(\kappa^{-3})$, while steps in z, y are a factor ϵ smaller;
- $T = O(\kappa)$ inner iterations.

Their stochastic variants also improve over ours by factor of ϵ^{-1}:

- $\tilde{O}(\epsilon^{-4})$ for the case of noisy f gradients,
- $\tilde{O}(\epsilon^{-6})$ for the case of noisy f and g gradients.

F^{3}SA: Variant with Momentum

Use mechanism of [Khanduri et al., 2021] to accelerate convergence in some cases.

Input: step sizes $\{\alpha_k, \gamma_k\}$; multiplier increment $\{\delta_k\}$; momentum-weight sequence $\{\eta_k\}$; step-size ratio ξ; initialization λ_0, x_0, y_0, z_0.

```
for $k = 0 ... K-1$ do
    $z_{k+1} \leftarrow z_k - \gamma_k \tilde{h}_{gz}^k$
    $y_{k+1} \leftarrow y_k - \alpha_k(\tilde{h}_{fy}^k + \lambda_k \tilde{h}_{gy}^k)$
    $x_{k+1} \leftarrow x_k - \xi\alpha_k(\tilde{h}_{fx}^k + \lambda_k(\tilde{h}_{gxy}^k - \tilde{h}_{gxz}^k))$
    $\lambda_{k+1} \leftarrow \lambda_k + \delta_k$
end for
```

$$\tilde{h}_{gz}^k := \nabla_y g(x_k, z_k; \phi_z^k) + (1 - \eta_k)\left(\tilde{h}_{gz}^{k-1} - \nabla_y g(x_{k-1}, z_{k-1}; \phi_z^k)\right).$$

Other quantities $\tilde{h}_{fy}$, $\tilde{h}_{gy}$, $\tilde{h}_{fx}$, $\tilde{h}_{gxy}$, $\tilde{h}_{gxz}$ are defined similarly, with the same sequence η_k.

Complexity of F^{3}SA

Set some parameters as in F^{2}SA:

$$\xi = 1, \quad \alpha_k = O((k+k_0)^{-a}), \quad \gamma_k = O((k+k_0)^{-c}),$$

where $0 \le c \le a \le 1$ and k_0 is an instance-dependent constant. But now

$$\delta_k = O(1/\kappa)\lambda_k, \quad \eta_k = k^{-2c}.$$

Complexity of F^{3}SA

Suppose F^{3}SA is run for K iterations and R is drawn randomly from $\{0, 1, \ldots, K\}$. Then

- If gradients of both f and g contain stochastic noise, then set $a = 3/5$ and $c = 2/5$ to get $\mathbb{E}\|\nabla F(x_R)\| \leq \tilde{O}(K^{-1/5})$.
- If gradient of f contains stochastic noise but gradients of g are exact, then set $a = 1/2$ and $c = 1/4$ to get $\mathbb{E}\|\nabla F(x_R)\| \leq \tilde{O}(K^{-1/4})$.
- If gradients of both f and g are exact, set $a = 1/3$ and $c = 0$ to get $\mathbb{E}\|\nabla F(x_R)\| \leq \tilde{O}(K^{-1/3})$.

Number of iterations to get $\mathbb{E}\|\nabla F(x_R)\| \leq \epsilon$ is ϵ^{-5}, ϵ^{-4}, and ϵ^{-3}, respectively.

Momentum gives no improvement in the exact-gradient case.

Can we get better complexity than ϵ^{-7}?

.... for the case in which gradients of both f and g have stochastic noise?

Suppose we have a "y^* oracle" that tells us what is $y^*(x_k)$ and $y^*_\lambda(x_k)$ for a given x_k, within ϵ.

- Recall that
$$\nabla \mathcal{L}^*_\lambda(x) = \nabla_x f(x,y) + \lambda(\nabla_x g(x,y^*_\lambda) - \nabla_x g(x,y^*)).$$
- Use unbiased estimate
$$\nabla \mathcal{L}^*_\lambda(x;\phi,\xi_1,\xi_2) = \nabla_x f(x,y;\phi) + \lambda(\nabla_x g(x,y^*_\lambda;\xi_1) - \nabla_x g(x,y^*;\xi_2)).$$
- Apply stochastic gradient directly to $\mathcal{L}^*_\lambda$.

Variance of gradient estimate is inflated for $\lambda = 1/\epsilon$:

$$\mathrm{Var}_{\phi,\xi_1,\xi_2}[\nabla \mathcal{L}^*_\lambda(x;\phi,\xi_1,\xi_2)] = \mathrm{Var}(\nabla_x f) + \lambda^2\,\mathrm{Var}(\nabla_x g).$$

Iterations for $\|\nabla \mathcal{L}^*_\lambda(x)\| \le \epsilon$ are $O(\text{variance}/\epsilon^4) = O(\epsilon^{-6})$.

Stochastic Smoothness

Require stochastic oracle for $\nabla_x g(x, y)$ to be Lipschitz in y:

$$\mathbb{E}_\xi \left[\|\nabla_x g(x, y_1; \xi) - \nabla_x g(x, y_2; \xi)\|^2\right] \leq L^2 \|y_1 - y_2\|^2.$$

- Control y_k and z_k so that $\|y_k - y^*_{\lambda_k}(x_k)\|$, $\|z_k - y^*(x_k)\|$, and $\|y^*_{\lambda_k}(x_k) - y^*(x_k)\|$ are all $O(1/\lambda_k)$.
- Recall

$$\nabla \mathcal{L}^*_{\lambda_k}(x_k; \phi, \xi) \approx \nabla_x f(x_k, y_k; \phi) + \lambda(\nabla_x g(x_k, y_k; \xi) - \nabla_x g(x_k, z_k; \xi)).$$

- Variance is reduced:

$$\mathrm{Var}_{\phi,\xi}[\nabla \mathcal{L}^*_{\lambda_k}(x_k; \phi, \xi)] = \mathrm{Var}(\nabla_x f(x_k, y_k; \phi)) + \lambda_k^2 \|y_k - z_k\|^2 = O(1).$$

Thus stochastic gradient applied to $\mathcal{L}^*_\lambda$ for $\lambda = O(1/\epsilon)$ requires $O(\epsilon^{-4})$ iterations.

Lower Bounds

Can we do better than $O(\epsilon^{-6})$ for the stochastic-gradient variant of F^{2}SA, without stochastic smoothness, under the "y^* oracle"?

For optimization of a (single-level) smooth nonconvex function, it's known that there is a worst case function $F(x)$ for which the complexity is $O(\epsilon^{-4})$. Turn this into a bilevel problem:

$$\min_{x,y} f(x,y) := y \quad \text{s.t. } y \in \arg\min_z g(x,z) := \frac{1}{2}(z - F(x))^2.$$

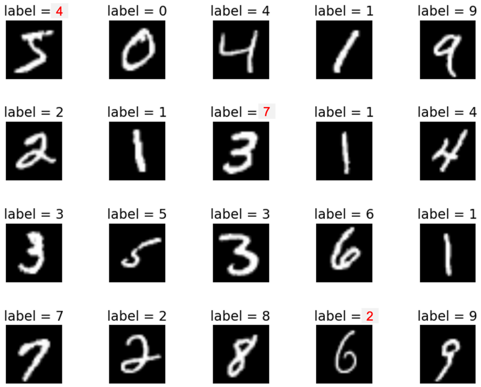
Note that $g(x,\cdot)$ is strongly convex with $\mu = 1$, f and g are smooth, and $y^*(x) = F(x)$.

But $\nabla_x g(x,y) = (y - F(x))\nabla F(x)$, where $\|y - F(x)\| = O(\epsilon)$ under the "y^* oracle."

This poor scaling of the gradient $\nabla_x g(x,y)$ relative to noise variance slows progress toward a stationary point x^*. Still require $O(\epsilon^{-6})$ iterations.

Experiment: Classification with Corrupted Labels

Training set has some labels corrupted. Use a validation set (with trusted labels) to adjust weights on the training examples to produce a reliable classifier.



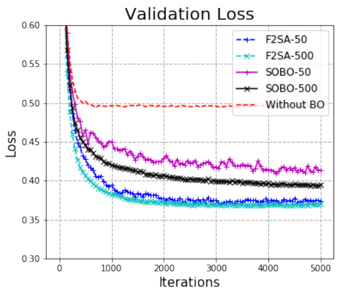
Experiment

Multiclass classification, labels y_i corrupted with some probability $p \in (0,1)$.

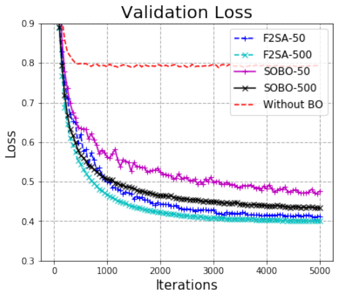
- Training set (with some corrupted labels): $(\tilde{x}_i, \tilde{y}_i)$, $i = 1, 2, \ldots, n$.
- Validation set (no corrupted labels): (x_i, y_i), $i = 1, 2, \ldots, m$.
- Training weights $\sigma(\lambda_i)$, $i = 1, 2, \ldots, m$, where $\sigma(\cdot) \in \mathbb{R}$ is logistic function and $\lambda_i \in \mathbb{R}$
- $l(\cdot)$ is multiclass logistic loss.
- $c =$ regularization parameter.

$$
\begin{aligned}
\min_{\lambda \in \mathbb{R}^n} \quad & \sum_{i=1}^{m} l(x_i, y_i; w^*) \\
\text{s.t.} \quad & w^* \in \arg\min_{w} \sum_{i=1}^{n} \sigma(\lambda_i) l(\tilde{x}_i, \tilde{y}_i; w) + c\|w\|^2.
\end{aligned}
$$

Experiment



(a)



(b)

Figure: Outer objective (validation) loss with label corruption rate: (a) $p = 0.1$, (b) $p = 0.3$.

- SOBO: algorithms using second-order oracles.
- $50/500 =$ batch sizes
- Without BO: No bilevel optimization to adjust weights.

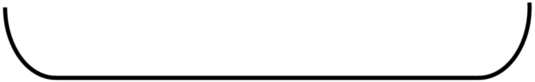
III. What if we don't have strongly convex lower level $g(x, \cdot)$?

$$\min_{x \in \mathbb{R}^{d_x}} F(x) := f(x, y^*(x)) \quad \text{s.t.} \quad y^*(x) \in \arg\min_{y \in \mathcal{Y}} g(x, y), \qquad \textbf{(P)}$$

- f and g are smooth;
- g may not be strongly convex, or even convex;
- lower-level constraint set $\mathcal{Y}$ is closed and convex.

Challenges:

1. Lower-level solution $y^*(x)$ may not be unique:



2. Hessians may not be invertible:

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

Questions: What is $\nabla F(x)$ in this situation? Do we still have $\nabla F(x) \approx \nabla \mathcal{L}^*_\lambda(x)$, for large λ?

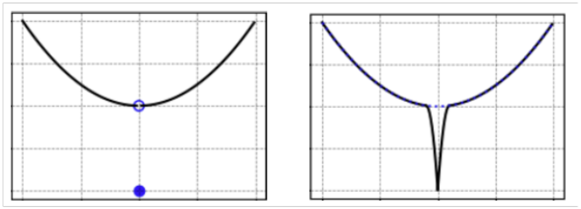
Elementary (Nasty) Example

Scalar x, y.

$$\min_{x\in[-1,1]} x^2 + y_*^2 \quad \text{subject to} \quad y_* \in \arg\min_{y\in[-1,1]} xy.$$

Leads to

$$F(x) = \begin{cases} x^2 + 1 & \text{if } x \neq 0 \\ 0 & \text{if } x = 0. \end{cases}$$



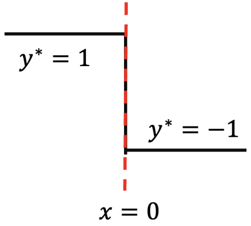


Figure: $F(x)$ (left), $\mathcal{L}_\lambda(x)$ (center), optimal y_* (right)

Sufficient Conditions for a Well-Defined Landscape

1. Lipschitz continuity of the lower-level solution set
   - Without this condition, finding a stationary point for x can be PPAD-Hard [Daskalakis et al., 2021]
   - Strong convexity / PL (Polyak-Lojasiewicz) for a certain combination of f and g implies solution-set continuity.
2. If the lower-level problem is constrained, also need nondegeneracy for the algebraic description of the (closed, convex) feasible set $\mathcal{Y}$.
   - constraint qualifications (LICQ)
   - strict complementarity.

Matching the Penalty Function $\mathcal{L}^*_\lambda(x)$ to the True Objective $F(x)$

Unconstrained lower level ($\mathcal{Y} = \mathbb{R}^{d_y}$). *In the below, think of σ as $1/\lambda$.*

Theorem

Consider solution sets $\mathcal{S}(x, \sigma)$ for the weighted problem

$$\mathcal{S}(x,\sigma) := \arg\min_{y \in \mathbb{R}^{d_y}} \sigma f(x,y) + g(x,y) =: h_\sigma(x,y)$$

If $\mathcal{S}(x,\sigma)$ is Lipschitz continuous in (x,σ) then

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

for any $y^* \in \mathcal{S}(x, 0^+)$. Moreover $\lim_{\sigma\downarrow 0} \nabla \mathcal{L}^*_{1/\sigma}(x) = \nabla F(x)$.

(† denotes pseudoinverse.) (Note: Don't require second-order sufficient conditions at (x^*, y^*).)

For the constrained lower-level problem, when nondegeneracy conditions hold at y^*, recover ∇F via a formula involving pseudoinverse of the KKT matrix.

Locally Approximated Landscape

We assume local continuity of the solution set $\mathcal{S}(x, \sigma)$ for the lower-level problem at x.

Define prox operator:

$$\mathrm{prox}_{\rho\phi}(y) = \arg\min_{z \in \Phi} \left\{ \rho\phi(z) + \frac{1}{2}\|z - y\|^2 \right\}.$$

(Assume that ρ is small enough to overcome nonconvexity in ϕ, e.g. $\rho^{-1} \gg \|\nabla^2 \phi(z)\|$.

(Uniform) Proximal Error Bound $\Rightarrow$ (Global) Solution set continuity:

$$\rho^{-1}\|y - \mathrm{prox}_{\rho h_\sigma(x,\cdot)}(y)\| \geq \mu \, \mathrm{dist}(y, \mathcal{S}(x, \sigma)), \tag{PL-σ}$$

(reminder: $h_\sigma(x, z) := \sigma f(x, z) + g(x, z)$.) (cf. Polyak-Lojasiewicz (PL)). Note that

$$y - \mathrm{prox}_{\rho h_\sigma(x,\cdot)}(y) = \rho \nabla_y h_\sigma(x, z)$$

for the minimizing z, making the connection to PL clear.

Locally Approximated Landscape

Strict convexity of $g(x, \cdot)$ implies (PL-σ), for small σ.

(PL-σ) can hold if any invariant subspace of $f(x, \cdot)$ contains the invariant subspace of $g(x, \cdot)$.

Recalling $\lambda \sim 1/\sigma$, we have (locally) the same kind of approximation result as in the strongly convex case:

$$|\mathcal{L}^*_{1/\sigma}(x) - F(x)| = O(\sigma/\mu) = O(\kappa\sigma), \quad \|\nabla \mathcal{L}^*_{1/\sigma}(x) - \nabla F(x)\| = O(\sigma/\mu^3) = O(\kappa^3\sigma).$$

(Uniform-Error-Bound holds for some neural network architectures [Frei and Gu, 2021].)

Formulation and Approach

Recall from (P-SADDLE) that the methods for the case of strongly convex $g(x, \cdot)$ could be viewed as gradient and stochastic gradient methods for the min-max problem

$$\min_{x\in\mathcal{X}, y\in\mathcal{Y}} \max_{z\in\mathcal{Y}} \sigma f(x,y) + (g(x,y) - g(x,z))$$

(with the substitution $\sigma = 1/\lambda$). We definition $h_\sigma(x,y) := \sigma f(x,y) + g(x,y)$, reformulate as

$$\min_{x\in\mathcal{X}, y\in\mathcal{Y}} \max_{z\in\mathcal{Y}} \{h_\sigma(x,y) - g(x,z)\} \Leftrightarrow \min_{x\in\mathcal{X}} \left\{ \min_{y\in\mathcal{Y}} h_\sigma(x,y) - \min_{z\in\mathcal{Y}} g(x,z) \right\}$$

But we can't count on convexity any more!

Note that y is a stationary point for the inner minimization $\min_{y\in\mathcal{Y}} h_\sigma(x,y)$ if y is a fixed point for the prox-operator

$$\mathrm{prox}_{\rho h_\sigma(x,\cdot)}(y) = \arg\max_{t\in\mathcal{Y}} \left\{ \rho h_\sigma(x,t) + \frac{1}{2}\|t-y\|^2 \right\}.$$

Formulation and Approach

For ρ sufficiently small (so that $\rho\|\nabla^2_{yy} h_\sigma(x,y)\| \ll 1$), the "prox" subproblem is convex.

Similarly, the z solving $\min_{z\in\mathcal{Y}} g(x,z)$ is a fixed point of $\operatorname{prox}_{\rho g(x,\cdot)}(\cdot)$.

Supposing that (y,z) are "optimal" for a given x, the stationarity condition for x is

$$x = P_{\mathcal{X}}\Big(x - \rho(\nabla_x h_\sigma(x,y) - \nabla_x g(x,z))\Big).$$

We say that (x^*,y^*,z^*) is an ϵ-stationary point of (P-SADDLE) if

$$\frac{1}{\rho}\|y^* - \operatorname{prox}_{\rho h_\sigma(x^*,\cdot)}(y^*)\| \leq \sigma\epsilon$$
$$\frac{1}{\rho}\|z^* - \operatorname{prox}_{\rho g(x^*,\cdot)}(z^*)\| \leq \sigma\epsilon$$
$$\frac{1}{\rho}\left\|x^* - P_X\Big(x^* - \rho(\nabla_x h_\sigma(x^*,y^*) - \nabla_x g(x^*,z^*))\Big)\right\| \leq \sigma\epsilon.$$

Formulation and Approach

These observations motivate an algorithm with three types of steps:

- gradient projection to solve the subproblem for z: $\mathrm{prox}_{\rho g(x,\cdot)}(z)$;
- gradient projection to solve the subproblem for y: $\mathrm{prox}_{\rho h_\sigma(x,\cdot)}(y)$;
- gradient projection to find a stationary point for x over $\mathcal{X}$.

As in the strongly convex case, there are several enhancements and variants:

- Possibly multiple iterations for y, z per iteration for x;
- Proximal smoothing of y, z iterates;
- Unbiased estimates of all the gradient terms $\nabla_x f$, $\nabla_y f$, $\nabla_x g$, $\nabla_y g$;
- Batching gradient estimates in x;
- Momentum assistance for x, y, z to accelerate convergence.

Two-Loop Algorithm

Input: total outer-loop iterations: K, step sizes: α_k, γ_k, proximal-smoothing parameters: $\beta_k \in (0,1]$, inner-loop iteration counts: T_k, outer-loop batch size: M_k, penalty parameters: σ_k, proximal parameter: ρ, initializations: $x_0 \in \mathcal{X}, y_0, z_0 \in \mathcal{Y}$

```
 1: Initialize $w_{y,0} = y_0$, $w_{z,0} = z_0$
 2: for $k = 0 \ldots K-1$ do
 3:     $u_0 \leftarrow w_{y,k}, v_0 \leftarrow w_{z,k}$
 4:     for $t = 0, \ldots, T_k - 1$ do
 5:         $v_{t+1} \leftarrow \mathrm{P}_{\mathcal{Y}}\left(v_t - \gamma_k(g_{wz}^{k,t} + \rho^{-1}(v_t - z_k))\right)$        compute $\mathrm{prox}_{\rho g(x_k,\cdot)}(z_k)$
 6:         $u_{t+1} \leftarrow \mathrm{P}_{\mathcal{Y}}\left(u_t - \gamma_k(\sigma_k f_{wy}^{k,t} + g_{wy}^{k,t} + \rho^{-1}(u_t - y_k))\right)$        compute $\mathrm{prox}_{\rho h_\sigma(x_k,\cdot)}(y_k)$
 7:     end for
 8:     $w_{y,k+1} \leftarrow u_T, w_{z,k+1} \leftarrow v_T$
 9:     $y_{k+1} \leftarrow (1-\beta_k)y_k + \beta_k w_{y,k+1}$        prox smoothing on $y$
10:     $z_{k+1} \leftarrow (1-\beta_k)z_k + \beta_k w_{z,k+1}$        prox smoothing on $z$
11:     $x_{k+1} \leftarrow \mathrm{P}_{\mathcal{X}}\left(x_k - \frac{\alpha_k}{M_k}\sum_{m=1}^{M_k}(\sigma_k f_x^{k,m} + g_{xy}^{k,m} - g_{xz}^{k,m})\right)$        gradient projection for $x$
12: end for
```

Convergence and Complexity

Set parameters

$$\rho \ll 1/l_{g,1}, \quad \alpha_k \sim \rho, \quad \beta_k \sim 1, \quad \sigma_k \sim (k+k_0)^{-1/3} \ (\text{with } \sigma_K = \epsilon),$$
$$\gamma_k \sim (k+k_0)^{-c}, \quad T_k \sim (k+k_0)^{c}, \quad M_k \sim (k+k_0)^{c}.$$

Suppose F^{2}SA is run for K iterations and R is drawn randomly from $\{0, 1, \dots, K\}$. Then w.p. at least $2/3$:

- If gradients of both f and g contain stochastic noise, then set $c = 4/3$ to get $\|\nabla \mathcal{L}^*_{1/\epsilon}(x_R)\| \le \epsilon$ in $O(\epsilon^{-7})$ gradient oracles.
- If gradient of f contains stochastic noise but gradients of g are exact, then set $c = 2/3$ to get $\mathbb{E}\|\nabla \mathcal{L}^*_{1/\epsilon}(x_R)\| \le \epsilon$ in $O(\epsilon^{-5})$ gradient oracles.
- If gradients of both f and g are exact, set $c = 0$ to get $\mathbb{E}\|\nabla \mathcal{L}^*_{1/\epsilon}(x_R)\| \le \epsilon$ in $O(\epsilon^{-3})$ gradient oracles.

Single-Loop Algorithm with Momentum

Input: total outer-loop iterations: K, step sizes: α_k, γ_k, proximal-smoothing parameters: $\beta_k \in (0,1]$ penalty parameters: σ_k, momentum schedulers: $\eta_k \in (0,1]$, proximal parameter: ρ, initializations: $x_0 \in \mathcal{X}, y_0, z_0 \in \mathcal{Y}$

```
1: Initialize $w_{y,0} = y_0$, $w_{z,0} = z_0$
2: **for** $k = 0 \dots K-1$ **do**
3:     $w_{z,k+1} \leftarrow \mathrm{P}_{\mathcal{Y}}\left(w_{z,k} - \gamma_k(\tilde{g}_{wz}^k + \rho^{-1}(w_{z,k} - z_k))\right)$        compute $\mathrm{prox}_{\rho g(x_k,\cdot)}(z_k)$
4:     $w_{y,k+1} \leftarrow \mathrm{P}_{\mathcal{Y}}\left(w_{y,k} - \gamma_k(\sigma_k \tilde{f}_{wy}^k + \tilde{g}_{wy}^k + \rho^{-1}(w_{y,k} - y_k))\right)$        compute $\mathrm{prox}_{\rho h_\sigma(x_k,\cdot)}(y_k)$
5:     $y_{k+1} \leftarrow (1-\beta_k)y_k + \beta_k w_{y,k+1}$        prox smoothing on $y$
6:     $z_{k+1} \leftarrow (1-\beta_k)z_k + \beta_k w_{z,k+1}$        prox smoothing on $z$
7:     $x_{k+1} \leftarrow \mathrm{P}_{\mathcal{X}}\left(x_k - \alpha_k\left(\sigma_k \tilde{f}_x^k + \tilde{g}_{xy}^k - \tilde{g}_{xz}^k\right)\right)$        gradient projection for $x$
8: **end for**
```

Momentum-assisted gradient estimators:

$$\tilde{g}_{wz}^k := \nabla_y g(x_k, w_{z,k}; \xi_{wz}^k) + (1-\eta_k)\left(\tilde{g}_{wz}^{k-1} - \nabla_y g(x_{k-1}, w_{z,k-1}; \xi_{wz}^k)\right),$$

with f_{wy}^k, $\tilde{g}_{wy}^k$, $\tilde{f}_x^k$, $\tilde{g}_{xy}^k$, $\tilde{g}_{xz}^k$ defined similarly.

Convergence and Complexity

Set parameters

$$\rho \ll 1/l_{g,1}, \quad \alpha_k \sim \rho(k+k_0)^{-a}, \quad \beta_k \sim (k+k_0)^{-a}, \quad \gamma_k \sim (k+k_0)^{-a},$$
$$\sigma_k \sim (k+k_0)^{-s} \;\; (\text{with } \sigma_K = \epsilon), \quad \eta_k \sim (k+k_0)^{-n}.$$

Suppose F^{2}SA is run for K iterations and R is drawn randomly from $\{0, 1, \ldots, K\}$. Then w.p. at least $2/3$:

- If gradients of both f and g contain stochastic noise, then set $a = 2/5$, $s = 1/5$, $n = 4/5$ to get $\|\nabla \psi_\epsilon(x_R)\| \leq \epsilon$ in $O(\epsilon^{-5})$ gradient oracles.
- If gradient of f contains stochastic noise but gradients of g are exact, then set $a = 1/4$, $s = 1/4$, $n = 1/2$ to get $\mathbb{E}\|\nabla \psi_\epsilon(x_R)\| \leq \epsilon$ in $O(\epsilon^{-4})$ gradient oracles.
- If gradients of both f and g are exact, set $a = 0$, $s = 1/3$, $n = 0$ to get $\mathbb{E}\|\nabla \psi_\epsilon(x_R)\| \leq \epsilon$ in $O(\epsilon^{-3})$ gradient oracles.

Conclusions

Bilevel optimization arises often in big-data problems, so is a topic of renewed interest.

We focus on methods of (stochastic) gradient type.

Connects to previous work on nonlinear constrained optimization problems that do not satisfy constraint qualifications, especially "mathematical programming with equilibrium constraints" (MPEC).

We do a nonasymptotic convergence analysis, rather than "traditional" global and local convergence analysis — following a trend of the past 15 years.

References I

Chen, L., Ma, Y., and Zhang, J. (2025).
Near-optimal nonconvex-strongly-convex bilevel optimization with fully first-order oracles.
Journal of Machine Learning Research, 26(109):1–56.

Daskalakis, C., Skoulakis, S., and Zampetakis, M. (2021).
The complexity of constrained min-max optimization.
In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1466–1478.

Frei, S. and Gu, Q. (2021).
A unified framework for the analysis of neural networks trained by gradient descent.
In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 7937–7949.

Khanduri, P., Zeng, S., Hong, M., Wai, H.-T., Wang, Z., and Yang, Z. (2021).
A near-optimal algorithm for stochastic bilevel optimization via double-momentum.
In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pages 30271–30283.

Kim, T., Wright, S. J., Bienstock, D., and Harnett, S. (2016).
Analyzing vulnerability of power systems with continuous optimization formulations.
IEEE Transactions on Network Science and Engineering, 3(3):132–146.