Stochastic Bilevel Optimization

Qingyang Zhu
@SI152: Numerical Optimization

Problem Formulation

$$\min_{x \in \mathbb{R}^p} \Phi(x) := f(x, y^*(x))$$

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

Strongly convex,
y*(x) is unique

- $f(x, y)$: outer-level loss; $g(x, y)$: inner-level loss
- $y^*(x)$: minimizer of inner-level loss $g(x, \cdot)$

Methods

Constrained single-level Optimization w/ KKT conditions

- ❏ Many constraints : not for ML ❌
- ❏ Constraints bring nonconvexity

Methods cont.

Efficient Gradient-based :

- Approximate Implicit Differentiation (AID)
- Iterative Differentiation (ITD)

$$\min_{x \in \mathbb{R}^p} \Phi(x) := f(x, y^*(x))$$
$$\text{s.t.} \quad y^*(x) = \arg\min_{y \in \mathbb{R}^q} g(x, y),$$

Use Hypergradient $\nabla \Phi(x_k) = \frac{\partial f(x_k, y^*(x_k))}{\partial x_k}$

To perform Gradient Descent

Problems of Hypergradient $\nabla \Phi(x_k) = \frac{\partial f(x_k, y^*(x_k))}{\partial x_k}$

$$\min_{x} \ \phi(x) = f(x, y^*(x))$$

$$\nabla_x \phi(x) = \nabla_x f(x, y^*(x)) + \nabla_{y^*} f(x, y^*(x)) \nabla_x y^*(x) \quad (1)$$

Since $y^*(x) = \underset{y}{\arg\min}\ g(x, y)$

$$\nabla_y g(x, y^*(x)) = 0$$

$$\nabla_x \nabla_y g(x, y^*(x)) + \nabla_y^2 g(x, y^*(x)) \nabla_x y^*(x) = 0 \quad (2)$$

Combine (1)、(2)

$$\nabla_x \phi(x) = \nabla_x f(x, y^*(x)) - \nabla_{y^*} f(x, y^*(x)) [\nabla_y^2 g(x, y^*(x))]^{-1} \times \nabla_x \nabla_y g(x, y^*(x))$$

The Inverse of Hessian: O(n^2+n^3)

AID Approximate Implicit Differentiation

$$\min_x \phi(x) = f(x, y^*(x))$$

$$\nabla_x \phi(x) = \nabla_x f(x, y^*(x)) + \nabla_{y^*} f(x, y^*(x)) \nabla_x y^*(x) \quad (1)$$

Since $y^*(x) = \arg\min_y g(x, y)$

$$\nabla_y g(x, y^*(x)) = 0$$

$$\nabla_x \nabla_y g(x, y^*(x)) + \nabla_y^2 g(x, y^*(x)) \nabla_x y^*(x) = 0 \quad (2)$$

Combine (1), (2)

$$\nabla_x \phi(x) = \nabla_x f(x, y^*(x)) - \boxed{\nabla_{y^*} f(x, y^*(x)) [\nabla_y^2 g(x, y^*(x))]^{-1}} \times \nabla_x \nabla_y g(x, y^*(x))$$

V

Conjugate Gradient -> v*

$\nabla_y^2 g(x_k, y_k^D)$ Positive definite

$$\min_v \frac{1}{2} v^T \nabla_y^2 g(x_k, y_k^D) v - v^T \nabla_y f(x_k, y_k^D)$$

Unconstrained quadratic optimization

$$\nabla_y^2 g(x_k, y_k^D) v = \nabla_y f(x_k, y_k^D)$$

A linear system

ITD Iterative Differentiation

$$\frac{\partial f(x_k, y_k^D(x_k))}{\partial x_k} \Longrightarrow \frac{\partial f(x_k, y^*(x_k))}{\partial x_k}$$

Compute via Back Propagation

AID & ITD Implementation

Algorithm 1 Bilevel algorithms via AID or ITD

1: **Input:** K, D, N, stepsizes α, β, initializations x_0, y_0, v_0.
2: **for** $k = 0, 1, 2, ..., K$ **do**
3: Set $y_k^0 = y_{k-1}^D$ if $k > 0$ and y_0 otherwise
4: **for** $t = 1, ..., D$ **do**
5: Update $y_k^t = y_k^{t-1} - \alpha \nabla_y g(x_k, y_k^{t-1})$
6: **end for**
7: Hypergradient estimation via
AID: 1) set $v_k^0 = v_{k-1}^N$ if $k > 0$ and v_0 otherwise
2) solve v_k^N from $\nabla_y^2 g(x_k, y_k^D)v = \nabla_y f(x_k, y_k^D)$ via N steps of CG starting from v_k^0
3) get Jacobian-vector product $\nabla_x \nabla_y g(x_k, y_k^D)v_k^N$ via automatic differentiation
4) $\widehat{\nabla}\Phi(x_k) = \nabla_x f(x_k, y_k^D) - \nabla_x \nabla_y g(x_k, y_k^D)v_k^N$

ITD: compute $\widehat{\nabla}\Phi(x_k) = \frac{\partial f(x_k, y_k^D)}{\partial x_k}$ via backpropagation
8: Update $x_{k+1} = x_k - \beta \widehat{\nabla}\Phi(x_k)$
9: **end for**

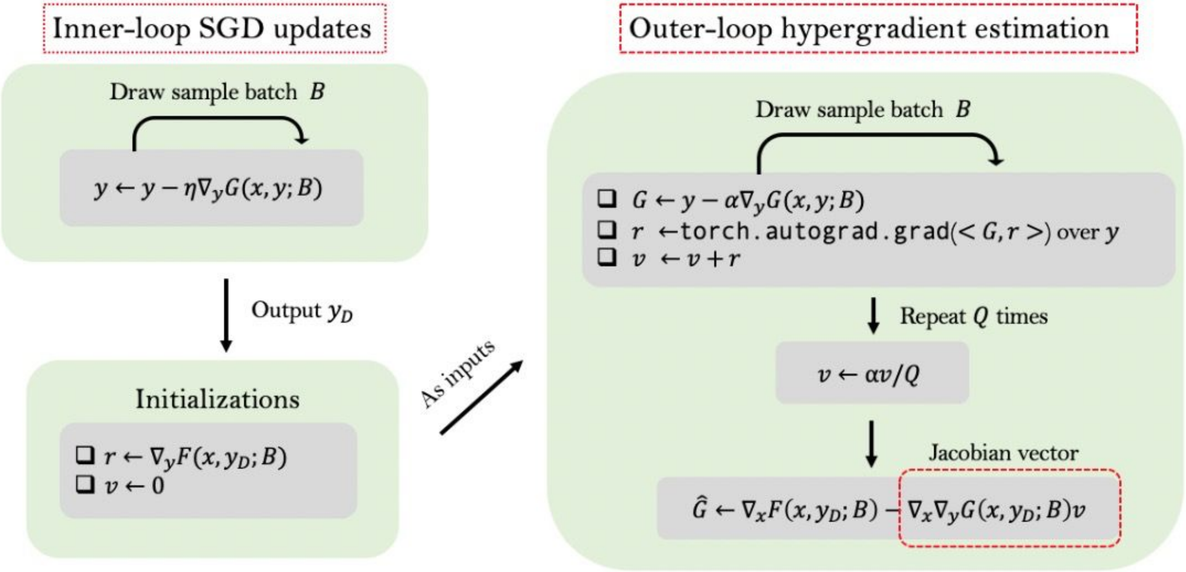
Both first use D gradient descent to find an approximate y*:= y_k^D

Stochastic Formulation

$$\min_{x \in \mathbb{R}^p} \Phi(x) = f(x, y^*(x)) = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} F(x, y^*(x); \xi_i) \\ \mathbb{E}_\xi \left[F(x, y^*(x); \xi)\right] \end{cases}$$

$$\text{s.t. } y^*(x) = \arg\min_{y \in \mathbb{R}^q} g(x, y) = \begin{cases} \frac{1}{m}\sum_{i=1}^{m} G(x, y; \zeta_i) \\ \mathbb{E}_\zeta \left[G(x, y; \zeta)\right] \end{cases} \tag{2}$$

StocBio



Inner-loop SGD updates
Draw sample batch B
$y \leftarrow y - \eta \nabla_y G(x, y; B)$
Output y_D
Initializations
$r \leftarrow \nabla_y F(x, y_D; B)$
$v \leftarrow 0$
As inputs
Outer-loop hypergradient estimation
Draw sample batch B
$G \leftarrow y - \alpha \nabla_y G(x, y; B)$
$r \leftarrow$ torch.autograd.grad($< G, r >$) over y
$v \leftarrow v + r$
Repeat Q times
$v \leftarrow \alpha v / Q$
Jacobian vector
$\hat{G} \leftarrow \nabla_x F(x, y_D; B) - \nabla_x \nabla_y G(x, y_D; B) v$

StocBio cont.

Algorithm 2 Stochastic bilevel optimizer (stocBiO)

```
Input: K, D, Q, stepsizes α and β, initializations x_0 and y_0.
for k = 0, 1, 2, ..., K do
    Set y_k^0 = y_{k-1}^D if k > 0 and y_0 otherwise
    for t = 1, ...., D do
        Draw a sample batch S_{t-1}
        Update y_k^t = y_k^{t-1} − α∇_y G(x_k, y_k^{t-1}; S_{t-1})
    end for
    Draw sample batches D_F, D_H and D_G
    Compute gradient v_0 = ∇_y F(x_k, y_k^D; D_F)
    Construct estimate v_Q via Algorithm 3 given v_0
    Compute ∇_x∇_y G(x_k, y_k^D; D_G) v_Q
    Compute gradient estimate \hat{∇}Φ(x_k) via eq. (6)
    Update x_{k+1} = x_k − β\hat{∇}Φ(x_k)
end for
```

Algorithm 3 Construct v_Q given v_0

```
Input: Integer Q, samples D_H = {B_j}_{j=1}^Q and constant η.
for j = 1, 2, ..., Q do
    Sample B_j and compute G_j(y) = y − η∇_y G(x, y; B_j)
end for
Set r_Q = v_0
for i = Q, ..., 1 do
    r_{i-1} = ∂(G_i(y) r_i)/∂y = r_i − η∇_y^2 G(x, y; B_i) r_i via
    automatic differentiation
end for
Return v_Q = η Σ_{i=0}^Q r_i
```

StocBio cont.

Algorithm 3 Construct v_Q given v_0

1: **Input:** Integer Q, samples $\mathcal{D}_H = \{\mathcal{B}_j\}_{j=1}^{Q}$ and constant η.
2: **for** $j = 1, 2, ..., Q$ **do**
3: Sample $\mathcal{B}_j$ and compute $G_j(y) = y - \eta \nabla_y G(x, y; \mathcal{B}_j)$
4: **end for**
5: Set $r_Q = v_0$
6: **for** $i = Q, ..., 1$ **do**
7: $r_{i-1} = \partial\big(G_i(y) r_i\big)/\partial y = r_i - \eta \nabla_y^2 G(x, y; \mathcal{B}_i) r_i$ via automatic differentiation
8: **end for**
9: Return $v_Q = \eta \sum_{i=0}^{Q} r_i$

$$\nabla_y^2 g(x_k, y_k^D) v = \nabla_y f(x_k, y_k^D)$$

Neumann Series

$$(\mathrm{Id} - T)^{-1} = \sum_{k=0}^{\infty} T^k,$$

Id: identical operator

We want A inverse:

$$T(\mathbf{x}) = (\mathbf{I} - \mathbf{A})\mathbf{x}$$

$$\mathbf{A}^{-1} \approx \sum_{i=0}^{n} (\mathbf{I} - \mathbf{A})^i$$

Performance

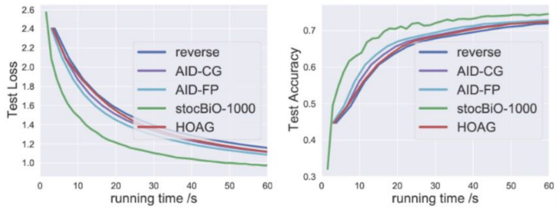
Fast Stochastic Bilevel Optimizer

- Lower complexity

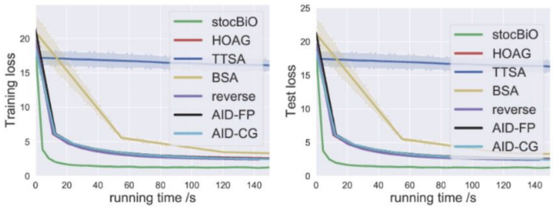
Algorithm	Gc(F,ϵ)	Gc(G,ϵ)	JV(G,ϵ)	HV(G,ϵ)
TTSA (Hong et al., 2020)	$\mathcal{O}(\text{poly}(\kappa)\epsilon^{-\frac{5}{2}})^*$	$\mathcal{O}(\text{poly}(\kappa)\epsilon^{-\frac{5}{2}})$	$\mathcal{O}(\text{poly}(\kappa)\epsilon^{-\frac{5}{2}})$	$\mathcal{O}(\text{poly}(\kappa)\epsilon^{-\frac{5}{2}})$
BSA (Ghadimi & Wang, 2018)	$\mathcal{O}(\kappa^6\epsilon^{-2})$	$\mathcal{O}(\kappa^9\epsilon^{-3})$	$\mathcal{O}\left(\kappa^6\epsilon^{-2}\right)$	$\widetilde{\mathcal{O}}\left(\kappa^6\epsilon^{-2}\right)$
stocBiO (this paper)	$\mathcal{O}(\kappa^5\epsilon^{-2})$	$\mathcal{O}(\kappa^9\epsilon^{-2})$	$\mathcal{O}\left(\kappa^5\epsilon^{-2}\right)$	$\widetilde{\mathcal{O}}\left(\kappa^6\epsilon^{-2}\right)$

ϵ: target accuracy; κ: condition number

- Fast convergence and strong efficiency:



Logistic regression on 20 Newsgroup



Data hyper-cleaning on MNIST

Thanks