Enhancing Bilevel Optimization with Single-Level Learning Techniques

Group G4: David Yang, Liuyuan Jiang, Meltem Tatli, Quan Xiao

Rensselaer Polytechnic Institute
CS 6962

December 11, 2023

Introduction

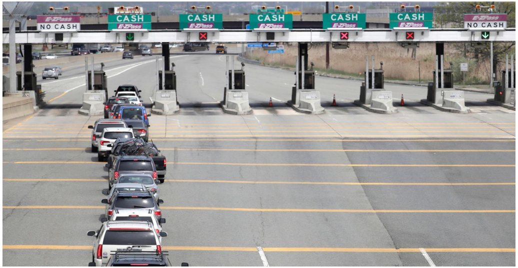
Bilevel Optimization

What is Bilevel Optimization?

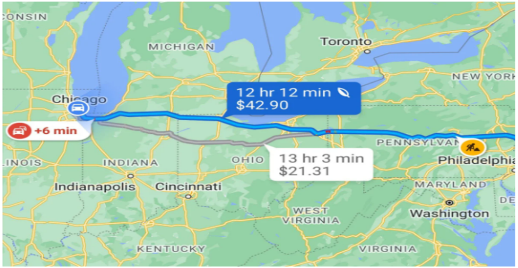
- Optimizing **two interconnected** optimization problems, where one problem is **nested** within another.

Why is it important?

- Many real world applications in transportation, economics, engineering etc.



(a) Government wishes to maximize tolls



(b) Travellers wish to minimize tolls

Bilevel Optimization

$$\min_x \; F(x) := f(x, y^*(x)), \quad \text{s.t.} \quad y^*(x) = \arg\min_y g(x, y) \tag{1}$$

- By chain rule,

$$\nabla F(x) = \nabla_x f(x, y^*(x)) + \nabla^\top y^*(x) \nabla_y f(x, y^*(x)). \tag{2}$$

- Gradient of bilevel objective can be rewritten as:

$$\nabla F(x) = \nabla_x f(x, y^*(x)) - \nabla^2_{xy} g(x, y^*(x)) \left[\nabla^2_{yy} g(x, y^*(x))\right]^{-1} \nabla_y f(x, y^*(x)). \tag{3}$$

- Can solve using gradient-based algorithm with Hessian inversion estimation, **but** requires second-order information -> computationally inefficient

Constrained Bilevel Optimization

- Instead, reformulate as:

$$\min_{x,y} f(x,y), \quad \text{s.t.} \quad g(x,y) - g^*(x) \leq 0. \tag{4}$$

- Then, we can minimize the Lagrangian function:

$$\mathcal{L}_\lambda(x,y) := f(x,y) + \lambda(g(x,y) - g^*(x)) \tag{5}$$

- This gradient is fully first-order, so we have:

$$\nabla \mathcal{L}_\lambda^*(x) = \nabla_x \mathcal{L}_\lambda(x, y_\lambda^*(x)), \quad \text{where} \quad y_\lambda^*(x) = \arg\min_y \mathcal{L}_\lambda(x,y) \tag{6}$$

- We can approximate second-order information $\nabla F(x)$ using fully first-order derivative $\nabla \mathcal{L}_\lambda^*(x)$

Literature Review

Nested Bilevel Methods

- **Classification:** Iterative differentiation (ITD) and approximation differentiation (AID).
- **ITD:** Solves lower-level problem iteratively. Nonasymptotic convergence rate established. Empirical efforts for memory reduction [1], [2].
- **AID:** Uses implicit function theorem and Hessian inversion estimation. Incorporates Neumann series, conjugate gradient, and kernel-based methods [3]–[5].
- **Advances:** Variance reduction, momentum methods [6], [7], warm-started algorithms [8], distributed and adaptive approaches [9], [10].

Fully First Order Bilevel Methods

- **Background:** Under-explored compared to second-order methods.
- **Developments:** Dynamic barrier gradient descent [11], Simple first-order method for strongly convex lower-level objectives [12], Penalized problem approach for nonconvex lower-level problems [13], Extension for constrained nonconvex upper and lower levels [14].
- **Note:** No incorporation of adaptive gradient schemes.

Methodology

Baseline algorithm: F^{2}SA

Recall the estimation of $\nabla F(x)$ is made through

$$\nabla F(x) \approx \nabla \mathcal{L}_\lambda^*(x) = \nabla_x \mathcal{L}_\lambda(x, y_\lambda^*(x)), \quad \text{where} \quad y_\lambda^*(x) = \arg\min_y \mathcal{L}_\lambda(x, y)$$

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

According to the Danskin theorem, we can further write $\nabla g^*(x) = \nabla_x g(x, y^*(x))$ and

$$\nabla \mathcal{L}_\lambda^*(x) = \nabla_x f(x, y_\lambda^*(x)) + \lambda(\nabla_x g(x, y_\lambda^*(x)) - \nabla_x g(x, y^*(x))) \quad \text{where} \quad y^*(x) = \arg\min_y g(x, y)$$

At each iteration, estimate $y_\lambda^*(x_k)$ and $y^*(x_k)$ by SGD output on $\mathcal{L}_{\lambda_k}(x_k, \cdot)$ and $g(x_k, \cdot)$.

Baseline algorithm: F^{2}SA

That is, at each iteration k, we initialize $z_{k,0} = z_k, y_{k,0} = y_k$ and execute T step GD with stepsize $\{\beta_k, \alpha_k\}$ in parallel,

$$z_{k,t+1} \leftarrow z_k - \beta \nabla_y g(x_k, z_{k,t}), \qquad y_{k,t+1} \leftarrow y_k - \alpha_k(\nabla_y f(x_k, y_{k,t}) + \lambda_k \nabla_y g(x_k, y_{k,t}))$$

Here $\lim_{t\to\infty} z_{k,t} = y^*(x_k)$ and $\lim_{t\to\infty} y_{k,t} = y^*_{\lambda_k}(x_k)$.

After that, we set $z_{k+1} = z_{k,T}, y_{k+1} = y_{k,T}$ and update x by GD with a stepsize ratio ξ,

$$x_{k+1} \leftarrow x_k - \xi\alpha_k(\nabla_x f(x_k, y_{k+1}) + \lambda_k(\nabla_x g(x_k, y_{k+1}) - \nabla_x f(x_k, z_{k+1})))$$

Baseline algorithm: F^{2}SA

Replacing the gradients by their stochastic unbiased estimators

$$h_{gz}^{k,t} := \nabla_y g\left(x_k, z_{k,t}; \phi_z^{k,t}\right), h_{fy}^{k,t} := \nabla_y f\left(x_k, y_{k,t}; \zeta_y^{k,t}\right), h_{gy}^{k,t} := \nabla_y g\left(x_k, y_{k,t}; \phi_y^{k,t}\right),$$
$$h_{gxy}^{k} := \nabla_x g\left(x_k, y_{k+1}; \phi_{xy}^{k}\right), h_{fx}^{k} := \nabla_x f\left(x_k, y_{k+1}; \zeta_x^{k}\right), h_{gxz}^{k} := \nabla_x g\left(x_k, z_{k+1}; \phi_{xz}^{k}\right)$$

and given a sequence of enlarging λ_k, we yield the F^{2}SA algorithm.

Baseline algorithm: F^{2}SA

Algorithm F^{2}SA

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

for $k = 0$ to $K - 1$ **do**

  $z_{k,0} \leftarrow z_k, y_{k,0} \leftarrow y_k$

  for $t = 0$ to $T - 1$ **do**

    $z_{k,t+1} \leftarrow z_{k,t} - \gamma_k h_{gz}^{k,t}, \quad y_{k,t+1} \leftarrow y_{k,t} - \alpha_k (h_{fy}^{k,t} + \lambda_k h_{gy}^{k,t})$

  end for

  $z_{k+1} \leftarrow z_{k,T}, y_{k+1} \leftarrow y_{k,T}$

  $x_{k+1} \leftarrow x_k - \xi \alpha_k (h_{fx}^k + \lambda_k (h_{gxy}^k - h_{gxz}^k))$

  $\lambda_{k+1} \rightarrow \lambda_k + \delta_k$

end for

Improved version: F^2SA with $Adam$.

- F^2SA: fully first-order method
  - consider no second-order information which introduces larger noises
- $Adam$:
  - approximate the second-order information using first-order
  - fixed the learning-rate decay issue of AdaGrad, being a popular method

Improved version: F^{2}SA with $Adam$.

Algorithm AdamGrad$(m_{t-1}, v_{t-1}, g_t, \beta_1, \beta_2, t)$

1: **Input:** Gradient estimator g_t, previous estimates m_{t-1}, v_{t-1}, parameters β_1, β_2
2: $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$
3: $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
4: $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$
5: $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$
6: **Output:** Updated estimates $(\hat{m}_t, \hat{v}_t)$

Improved version: F^{2}SA with $Adam$.

Algorithm F^{2}SA with $Adam$

```
Input: step sizes: {α_k, γ_k}, multiplier difference sequence: {δ_k}, inner-loop iteration T, initializations:
    λ_0, x_0, y_0, z_0, m_{x,−1} ← 0, v_{x,−1} ← 0, parameters: β_1, β_2, ϵ, ξ
for k = 0 to K − 1 do
    z_{k,0} ← z_k, y_{k,0} ← y_k
    m_{y,−1} ← 0, v_{y,−1} ← 0, m_{z,−1} ← 0, v_{z,−1} ← 0
    for t = 0 to T − 1 do
        Update (m_{z,t}, v_{z,t}) = AdamGrad(m_{z,t−1}, v_{z,t−1}, h_{gz}^{k,t}, β_1, β_2, t)
        z_{k,t+1} ← z_{k,t} − γ_k m_{z,t} / (sqrt(v_{z,t}) + ϵ)
        Update (m_{y,t}, v_{y,t}) = AdamGrad(m_{y,t−1}, v_{y,t−1}, h_{fy}^{k,t} + λ_k h_{gy}^{k,t}, β_1, β_2, t)
        y_{k,t+1} ← y_{k,t} − α_k m_{y,t} / (sqrt(v_{y,t}) + ϵ)
    end for
    z_{k+1} ← z_{k,T}, y_{k+1} ← y_{k,T}
    Update (m_{x,k}, v_{x,k}) = AdamGrad(m_{x,k−1}, v_{x,k−1}, h_{fx}^k + λ_k (h_{gxy}^k − h_{gxz}^k), β_1, β_2, k)
    x_{k+1} ← x_k − ξ α_k m_{x,k} / (sqrt(v_{x,k}) + ϵ)
    λ_{k+1} ← λ_k + δ_k
end for
```

Experiment Results

Data Hyper-Cleaning

- **Goal:** Train a classifier within a corrupted environment that can generalize well to clean, unseen data.
- Labels in the training dataset is substituted with a random class number according to a specified corruption rate p_c
- x is a vector being trained to label the noisy data and y is the model weight and bias, the objective is given by

$$\begin{aligned} &\min_x \frac{1}{|\mathcal{D}_{\text{val}}|} \sum_{(a_i,b_i)\in\mathcal{D}_{\text{val}}} \mathrm{CE}\left(y^*(x); a_i, b_i\right) \\ &\text{s.t. } \; y^*(x) = \arg\min_y \frac{1}{|\mathcal{D}_{\text{tr}}|} \sum_{(a_i,b_i)\in\mathcal{D}_{\text{tr}}} [\sigma(x)]_i \mathrm{CE}\left(y; a_i, b_i\right) + \frac{r}{2}\|y\|^2 \end{aligned}$$

Data Hyper-Cleaning: Hyper parameter Search

Parameters	F^2SA	F^2SA with Adam
$\{\gamma_k, \alpha_k\}$	{1, 0.5, 0.1, 0.05, 0.01, 0.005}	{0.01, 0.005, 0.001, 0.0009, 0.0005}
ξ	{0.5, 1, 5}	{0.5, 1, 5}
δ_k	{0.01, 0.1, 0.5, 1}	{0.01, 0.1, 0.5, 1}
β_1	/	{0.9, 0.85, 0.8, 0.75, 0.7}
β_2	/	{0.999, 0.99, 0.98, 0.95,0.9, 0.85}
batch size	{16, 32, 64}	{16, 32, 64}

Table: Search grid for parameters in F^2SA and F^2SA with Adam.

Data Hyper-Cleaning: Selected Hyper parameters

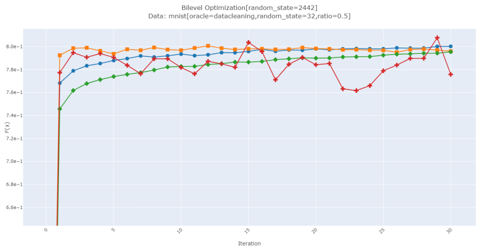
Parameters	F^2SA		F^2SA with Adam	
	MNIST	FashionMNIST	MNIST	FashionMNIST
$\{\gamma_k, \alpha_k\}$	0.05	0.1	0.0009	0.001
ξ	1	1	5	1
δ_k	0.1	0.1	0.1	0.1
β_1	/	/	0.9	0.9
β_2	/	/	0.99	0.99
batch size	64	64	64	64

Table: Selected parameters in F^2SA and F^2SA with Adam on MNIST and FashionMNIST.

Data Hyper-Cleaning Results (corruption rate $p_c = 0.5$)

- Compared F2SA with Adam with the other baseline algorithms
- Optimum step size for F2SA is larger compared to F2SA with Adam
- F^2SA with Adam converges faster.

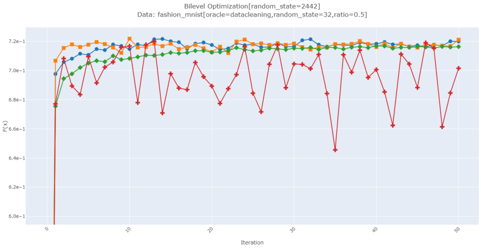
MNIST Dataset



(a) Test accuracy v.s. iteration comparison plot of F^2SA with Adam, F^2SA and StocBiO, SABA.

Data Hyper-Cleaning Results (corruption rate $p_c = 0.5$)

FashionMNIST Dataset



(a) Test accuracy v.s. iteration comparison plot of F^2SA with Adam, F^2SA and StocBiO, SABA.

Data Hyper-Cleaning Results (corruption rate $p_c = 0.7$)

FashionMNIST Dataset

(a) Test accuracy v.s. iteration comparison plot of F^2SA with Adam, F^2SA and StocBiO, SABA.

Regularization Selection

- **Goal:** Optimizing regularization coefficient x, which is used in training a model y on the training set, such that the learned model achieves the low risk on the validation set.
- Let $\ell(y; a_i, b_i)$ denote the loss of the model y on datum a_i and label b_i, and $\mathcal{D}_{\text{val}}$ and $\mathcal{D}_{\text{tr}}$ denote, respectively, the training and validation datasets.

$$\min_x \frac{1}{|\mathcal{D}_{\mathsf{val}}|} \sum_{(a_i,b_i)\in\mathcal{D}_{\mathsf{val}}} \ell(y^*(x); a_i, b_i)$$

$$\text{s.t.} \quad y^*(x) = \arg\min_y \frac{1}{|\mathcal{D}_{\mathsf{tr}}|} \sum_{(a_i,b_i)\in\mathcal{D}_{\mathsf{tr}}} \ell(y; a_i, b_i) + \sum_{i=1}^{|\mathcal{D}_{\mathsf{tr}}|} \exp(x_i)\|y_i\|^2. \tag{7}$$

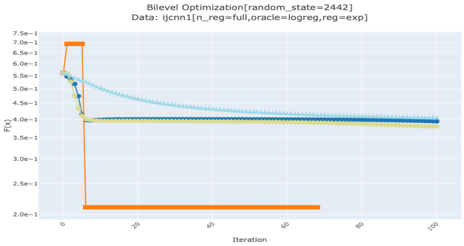
Regularization Selection Results: Hyper Parameter Selection

Parameters	F^2SA	F^2SA with Adam
$\{\gamma_k, \alpha_k\}$	0.1	0.05
ξ	1	1
δ_k	0.1	0.1
β_1	/	0.9
β_2	/	0.85
batch size	64	32

Table: Selected parameters in F^2SA and F^2SA with Adam on Ijcnn1.

Regularization Selection Results

- Ijcnn1 Dataset
- $\ell(y; a_i, b_i)$ is chosen as the cross entropy loss



(a) Test accuracy v.s. iteration comparison plot of F^2SA with Adam, F^2SA and StocBiO, SABA.

Conclusions

Conclusions

- Boost a first order bilevel method F^2SA by the adaptive gradient method Adam
- Conducted data hyper-clearning and regularization selection experiments which demonstrated the algorithm's accelerated convergence and stability with respect to parameters and settings
- Faster convergence in data hyper-cleaning task compared to base algorithms
- Achieve lowest objective value for regularization selection task

References

D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *Proc. International Conference on Machine Learning*, Lille, France, 2015, E. Grefenstette, B. Amos, D. Yarats, *et al.*, "Generalized inner loop meta-learning," *arXiv preprint arXiv:1910.01727*, 2019, S. Ghadimi and M. Wang, "Approximation methods for bilevel programming," *arXiv preprint arXiv:1802.02246*, 2018, T. Chen, Y. Sun, and W. Yin, "Closing the gap: Tighter analysis of alternating stochastic gradient methods for bilevel problems," *Proc. Advances in Neural Information Processing Systems*, 2021, R. Hataya and M. Yamada, "Nyström method for accurate and scalable implicit differentiation," in *Proc. International Conference on Artificial Intelligence and Statistics*, Valencia, Spain, 2023, P. Khanduri, S. Zeng, M. Hong, *et al.*, "A near-optimal algorithm for stochastic bilevel optimization via double-momentum," in *Proc. Advances in Neural Information Processing Systems*, virtual, 2021, J. Yang, K. Ji, and Y. Liang, "Provably faster algorithms for bilevel optimization," in *Proc. Advances in Neural Information Processing Systems*, virtual, 2021, J. Kwon, D. Kwon, S. Wright, *et al.*, "A fully first-order method for stochastic bilevel optimization," in *Proc. International Conference on Machine Learning*, Honolulu, HI, 2023

References

M. Arbel and J. Mairal, "Amortized implicit differentiation for stochastic bilevel optimization," in *Proc. International Conference on Learning Representations*, virtual, 2022, D. A. Tarzanagh, M. Li, C. Thrampoulidis, *et al.*, "FEDNEST: Federated bilevel, minimax, and compositional optimization," in *Proc. International Conference on Machine Learning*, Baltimore, MD, 2022, C. Fan, G. Choné-Ducasse, M. Schmidt, *et al.*, "Bisls/sps: Auto-tune step sizes for stable bi-level optimization," in *Proc. Advances in Neural Information Processing Systems*, New Orleans, LA, 2023, B. Liu, M. Ye, S. Wright, *et al.*, "Bome! bilevel optimization made easy: A simple first-order approach," in *Proc. Advances in Neural Information Processing Systems*, New Orleans, LA, 2022, J. Kwon, D. Kwon, S. Wright, *et al.*, "A fully first-order method for stochastic bilevel optimization," in *Proc. International Conference on Machine Learning*, Honolulu, HI, 2023, H. Shen and T. Chen, "On penalty-based bilevel gradient descent method," in *Proc. International Conference on Machine Learning*, Honolulu, HI, 2023, J. Kwon, D. Kwon, S. Wright, *et al.*, "On penalty methods for nonconvex bilevel optimization and first-order stochastic approximation," *arXiv preprint arXiv:2309.01753*, 2023

Thanks for your attention!

Questions?