A framework for bilevel optimization that enables stochastic and global variance reduction algorithms

Thomas Moreau INRIA Saclay

Joint work with M. Dagréou, P. Ablin, S. Vaiter and Z. Ramzi

Bi-level problem: Optimization problem with two levels

$$\min_{\lambda} h(\lambda) = F(\lambda, \theta^*(\lambda))$$

Outer function

$$s.t. \quad \theta^*(\lambda) = \operatorname*{argmin}_{\theta} G(\lambda, \theta)$$

Value function

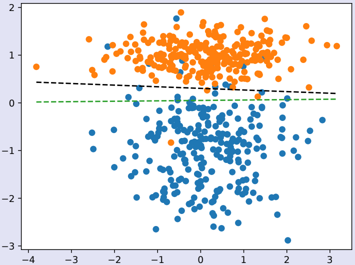
Inner function/Problem

Goal: Optimize the value function h whose value depends on the result of another optimization problem.

Selecting the best model:

- G is the training loss and θ are the parameters of the model.
- Select the hyper-parameter λ to get the best validation loss F.

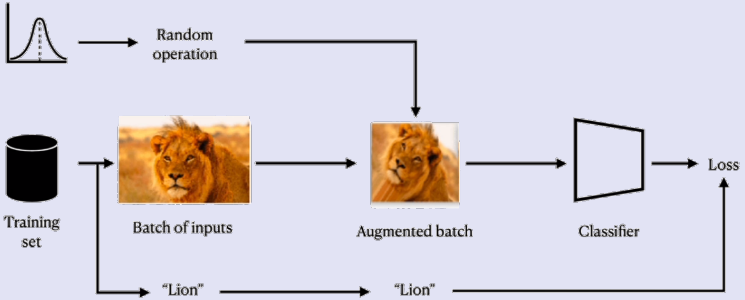
Hyperparameter optimization: λ is a regularization parameter:



Selecting the best model:

- G is the training loss and θ are the parameters of the model.
- Select the hyper-parameter λ to get the best validation loss F.

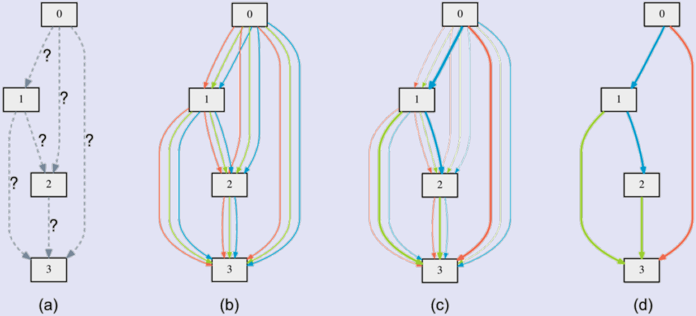
Data augmentation: λ parametrizes the transformations distribution.



Selecting the best model:

- G is the training loss and θ are the parameters of the model.
- Select the hyper-parameter λ to get the best validation loss F.

Neural Architecture Search: λ parametrizes the architecture.



Deep Equilibrium Network:

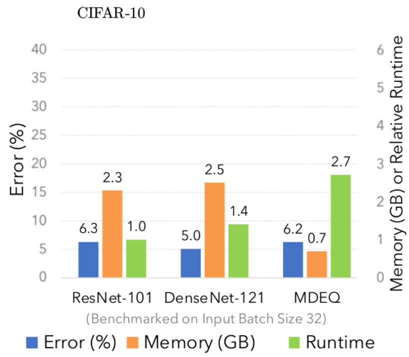
$$\begin{cases} \min_\lambda h(\lambda) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(y_i, \theta^*(X_i,\lambda)) \\ s.t. \quad \theta^*(X_i,\lambda) = g_\lambda(\theta^*(X_i,\lambda)) \end{cases}$$

Output of the network is the root of $G(\theta,\lambda) = \theta - g_\lambda(\theta) = 0$.

- **Mimic infinite depth:**

$$\theta^{(t+1)} = g_\lambda(\theta^{(t)}) \quad t \to \infty \ .$$

- **Efficient memory**
- **Slow runtime**



Solving bi-level optimization

Black box methods: Take $\{\lambda_k\}_k$ and compute $\min_k h(\lambda_k)$

- Grid-Search
- Random-Search
- Bayesian-Optimization

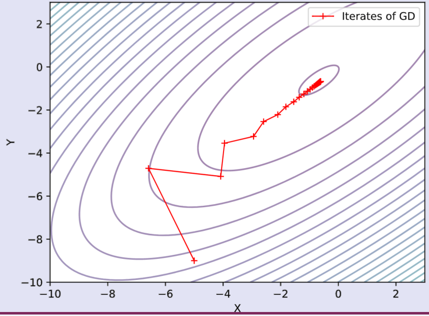
$\Rightarrow$ Do not scale well with the dimension

First order methods: Gradient descent on h

Iterate in the steepest direction:

$$\lambda^{t+1} = \lambda^t - \rho^t \nabla h(\lambda)$$

- Gradient $\nabla h(\lambda) = \frac{d\ F(\lambda, \theta^*(\lambda))}{d\ \lambda}$
- Step size ρ^t.



Value function definition:

$$h(\lambda) = F(\lambda, \theta^*(\lambda))$$

Value function gradient:

$$\nabla h(\lambda) = \nabla_1 F(\lambda, \theta^*) - \nabla^2_{21} G(\lambda, \theta^*) \Big[\nabla^2_{22} G(\lambda, \theta^*)\Big]^{-1} \nabla_2 F(\lambda, \theta^*)$$

Computing the gradient of h

Value function definition:

$$h(\lambda) = F(\lambda, \theta^*(\lambda))$$

Value function gradient:

$$\nabla h(\lambda) = \nabla_1 F(\lambda, \theta^*) - \nabla^2_{21} G(\lambda, \theta^*) \left[\nabla^2_{22} G(\lambda, \theta^*)\right]^{-1} \nabla_2 F(\lambda, \theta^*)$$

- Need to compute the solution of the inner

Computing the gradient of h

Value function definition:

$$h(\lambda) = F(\lambda, \theta^*(\lambda))$$

Value function gradient:

$$\nabla h(\lambda) = \nabla_1 F(\lambda, \theta^*) - \nabla^2_{21} G(\lambda, \theta^*) \left[\nabla^2_{22} G(\lambda, \theta^*)\right]^{-1} \nabla_2 F(\lambda, \theta^*)$$

- Need to compute the solution of the inner
- Need to solve a $p \times p$ linear system

$$v^*(\lambda) = \left[\nabla^2_{22} G(\lambda, \theta^*)\right]^{-1} \nabla_2 F(\lambda, \theta^*)$$

Approximate bilevel optimization

Approximating the Hypergradient

Two-loops approaches

Idea: Approximate $\theta^*(\lambda^t)$ and $v^*(\lambda^t)$ at each iteration.

Two-loops approaches

Idea: Approximate $\theta^*(\lambda^t)$ and $v^*(\lambda^t)$ at each iteration.

- Compute θ^t such that $\|\theta^t - \theta^*(\lambda^t)\|_2 \leq \epsilon_t$,
  iterative solver *e.g.* L-BFGS
- Compute hypergradient $g^t = \nabla_1 F(\lambda^t, \theta^t) + \nabla^2_{12} G(\lambda^t, \theta^t) v^t$ with $v^t \approx \left[\nabla^2_{22} G(\lambda^t, \theta^t)\right]^{-1} \nabla_2 F(\lambda^t, \theta^t)$ with error ϵ_t,
  linear system solver *e.g.* CG
- Update λ^t with $\lambda^{t+1} = \lambda^t - \rho^t g^t$.

Two-loops approaches

Idea: Approximate $\theta^*(\lambda^t)$ and $v^*(\lambda^t)$ at each iteration.

- Compute θ^t such that $\|\theta^t - \theta^*(\lambda^t)\|_2 \leq \epsilon_t$,

  iterative solver *e.g.* L-BFGS

- Compute hypergradient $g^t = \nabla_1 F(\lambda^t, \theta^t) + \nabla^2_{12} G(\lambda^t, \theta^t) v^t$ with $v^t \approx \left[\nabla^2_{22} G(\lambda^t, \theta^t)\right]^{-1} \nabla_2 F(\lambda^t, \theta^t)$ with error ϵ_t,

  linear system solver *e.g.* CG

- Update λ^t with $\lambda^{t+1} = \lambda^t - \rho^t g^t$.

Theorem (Two-loops Convergence ; Pedregosa 2016)

If $\sum_t \epsilon_t < \infty$ and the step-sizes are chosen appropriatly, then the algorithm converges to a stationary point of h i.e.

$$\|\nabla h(\lambda^t)\|_2 \to 0 \ .$$

Further linear system approximation v^*

Linear system solution $v^*(\lambda^t)$ is a by-product.

$\Rightarrow$ Avoid computing it as much as possible.

Proposed Methods:

- Conjugate Gradient
- Jacobian-Free method

$$\nabla^2_{22} G(\lambda^t, \theta^t) \approx Id$$

- SHINE
- Algorithm unrolling (*backprop.*)
- Neumann iterations

$$\nabla^2_{22} G(\lambda^t, \theta^t)^{-1} \approx \sum_k (Id - \nabla^2_{22} G(\lambda^t, \theta^t))^k$$

[Pedregosa 2016, Lorraine et al. 2020, Luketina et al. 2016, ...]

Quasi Newton 101:

Solving $\theta^*(\lambda) = \text{argmin}_\theta\, G(\lambda, \theta)$

Newton Method

$$\theta^{t+1} = \theta^t - \left[\nabla^2 G(\theta^t)\right]^{-1} \nabla G(\theta^t)$$

Quasi-Newton Method

$$\theta^{t+1} = \theta^t - B_t^{-1} \nabla G(\theta^t)$$

B_t^{-1}: low-rank approx. of $\nabla^2 G(\theta^t)^{-1}$.

Quasi Newton 101:

$$\text{Solving } \theta^*(\lambda) = \operatorname{argmin}_\theta G(\lambda, \theta)$$

Newton Method

$$\theta^{t+1} = \theta^t - \left[\nabla^2 G(\theta^t)\right]^{-1} \nabla G(\theta^t)$$

Quasi-Newton Method

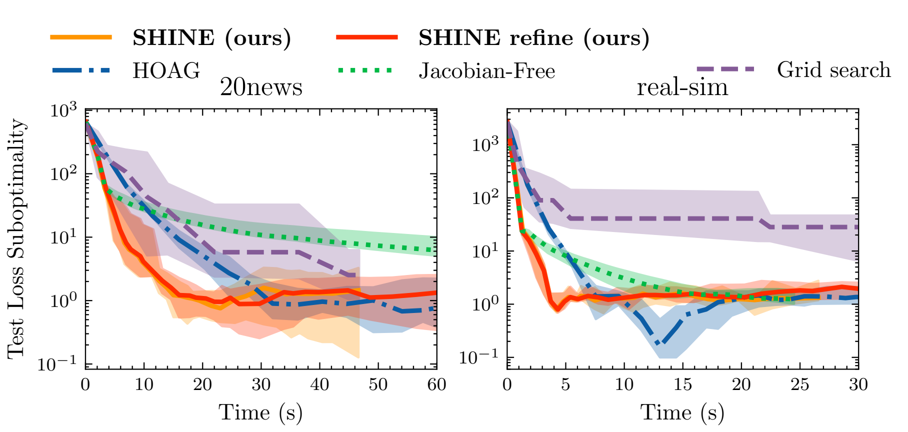
$$\theta^{t+1} = \theta^t - B_t^{-1} \nabla G(\theta^t)$$

B_t^{-1}: low-rank approx. of $\nabla^2 G(\theta^t)^{-1}$.

$\Rightarrow$ **SHINE** propose to use B_t^{-1} to approximate $v^*(\lambda^t)$

$$g^t = \nabla_1 F(\lambda^t, \theta^t) + \nabla^2_{12} G(\lambda^t, \theta^t) B_t^{-1} \nabla_2 F(\lambda^t, \theta^t)$$

Logistic Regression with ℓ_2-regularisation on 2 datasets:



Stochastic Bilevel Optimization

Stochastic bilevel optimization

$$F(\lambda, \theta) = \frac{1}{m} \sum_{j=1}^{m} F_j(\lambda, \theta), \quad G(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} G_i(\lambda, \theta)$$

Stochastic bilevel optimization

$$F(\lambda, \theta) = \frac{1}{m} \sum_{j=1}^{m} F_j(\lambda, \theta), \quad G(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} G_i(\lambda, \theta)$$

Stochastic updates:

- Compute θ^t with SGD,
- Compute stochastic $g^t = \nabla_1 F_j(\lambda^t, \theta^t) + \nabla^2_{12} G_i(\lambda^t, \theta^t) v^t$ with $v^t \approx \left[\nabla^2_{22} G_i(\lambda^t, \theta^t)\right]^{-1} \nabla_2 F_j(\lambda^t, \theta^t)$,
- Update λ^t with g^t.

Stochastic bilevel optimization

$$F(\lambda, \theta) = \frac{1}{m} \sum_{j=1}^{m} F_j(\lambda, \theta), \quad G(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} G_i(\lambda, \theta)$$

Stochastic updates:

- Compute θ^t with SGD,
- Compute stochastic $g^t = \nabla_1 F_j(\lambda^t, \theta^t) + \nabla^2_{12} G_i(\lambda^t, \theta^t) v^t$ with $v^t \approx \left[\nabla^2_{22} G_i(\lambda^t, \theta^t)\right]^{-1} \nabla_2 F_j(\lambda^t, \theta^t)$,
- Update λ^t with g^t.

Problem:

$$\left[\sum_{i=1}^{n} \nabla^2_{22} G_i(\lambda, \theta^*(\lambda))\right]^{-1} \neq \sum_{i=1}^{n} \left[\nabla^2_{22} G_i(\lambda, \theta^*(\lambda))\right]^{-1}$$

Stochastic linear system resolution

Stochastic approximation of $v^t = \left[\nabla^2_{22} G(\lambda^t, \theta^t)\right]^{-1} \nabla_2 F_j(\lambda^t, \theta^t)$.

- Neumann approximations [Ghadimi et al. 2018, Ji et al. 2021]:

$$v^t \approx \eta \sum_{q=0}^{Q} \prod_{k=0}^{q} \left(I - \eta \nabla^2_{22} G_{i_k}(\lambda^t, \theta^t)\right) \nabla_2 F_j(\lambda^t, \theta^t)$$

Stochastic linear system resolution

Stochastic approximation of $v^t = \left[\nabla^2_{22} G(\lambda^t, \theta^t)\right]^{-1} \nabla_2 F_j(\lambda^t, \theta^t)$.

- Neumann approximations [Ghadimi et al. 2018, Ji et al. 2021]:

$$v^t \approx \eta \sum_{q=0}^{Q} \prod_{k=0}^{q} \left(I - \eta \nabla^2_{22} G_{i_k}(\lambda^t, \theta^t)\right) \nabla_2 F_j(\lambda^t, \theta^t)$$

- Stochastic Gradient Descent [Grazzi et al. 2021, Arbel et al. 2021]

$$v^t \in \operatorname*{argmin}_{v \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \langle \nabla^2_{22} G_i(\lambda^t, \theta^t) v, v \rangle + \frac{1}{n} \sum_{j=1}^{m} \langle \nabla_2 F_j(\lambda^t, \theta^t), v \rangle$$

Stochastic linear system resolution

Stochastic approximation of $v^t = \left[\nabla^2_{22} G(\lambda^t, \theta^t)\right]^{-1} \nabla_2 F_j(\lambda^t, \theta^t)$.

- Neumann approximations [Ghadimi et al. 2018, Ji et al. 2021]:

$$v^t \approx \eta \sum_{q=0}^{Q} \prod_{k=0}^{q} \left(I - \eta \nabla^2_{22} G_{i_k}(\lambda^t, \theta^t)\right) \nabla_2 F_j(\lambda^t, \theta^t)$$

- Stochastic Gradient Descent [Grazzi et al. 2021, Arbel et al. 2021]

$$v^t \in \operatorname*{argmin}_{v \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \langle \nabla^2_{22} G_i(\lambda^t, \theta^t) v, v \rangle + \frac{1}{n} \sum_{j=1}^{m} \langle \nabla_2 F_j(\lambda^t, \theta^t), v \rangle$$

Stochastic linear system resolution

Stochastic approximation of $v^t = \left[\nabla^2_{22} G(\lambda^t, \theta^t)\right]^{-1} \nabla_2 F_j(\lambda^t, \theta^t)$.

- Neumann approximations [Ghadimi et al. 2018, Ji et al. 2021]:

$$v^t \approx \eta \sum_{q=0}^{Q} \prod_{k=0}^{q} \left(I - \eta \nabla^2_{22} G_{i_k}(\lambda^t, \theta^t)\right) \nabla_2 F_j(\lambda^t, \theta^t)$$

- Stochastic Gradient Descent [Grazzi et al. 2021, Arbel et al. 2021]

$$v^t \in \operatorname*{argmin}_{v \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \langle \nabla^2_{22} G_i(\lambda^t, \theta^t) v, v \rangle + \frac{1}{n} \sum_{j=1}^{m} \langle \nabla_2 F_j(\lambda^t, \theta^t), v \rangle$$

$\Rightarrow$ Still need to solve 2 optimization problems for every updates on the outer variable.

One-loop Approaches

Toward linear updates

References

- Dagréou, M., Ablin, P., Vaiter, S., and **TM** (2022). A framework for bilevel optimization that enables stochastic and global variance reduction algorithms, In *NeurIPS*
- Dagréou, M., **TM**, Vaiter, S., and Ablin, P. (2023). A Lower Bound and a Near-Optimal Algorithm for Bilevel Empirical Risk Minimization

Main idea

Three variables to maintain:

- $\theta^t \rightarrow$ solution to the inner problem
- $v^t \rightarrow$ solution of the linear system
- $\lambda^t \rightarrow$ solution of the outer problem

Idea: evolve in θ^t, v^t and λ^t at the same time following well chosen directions.

Update directions:

$$D_\theta(\theta, v, \lambda) = \nabla_2 G(\lambda, \theta) \quad \text{gradient step toward } \theta^*(\lambda)$$

Update directions:

$$D_\theta(\theta, v, \lambda) = \nabla_2 G(\lambda, \theta) \quad \text{gradient step toward } \theta^*(\lambda)$$

$$D_v(\theta, v, \lambda) = \nabla^2_{22} G(\lambda, \theta) v + \nabla_2 F(\lambda, \theta)$$

$$\text{gradient step toward } -\left[\nabla^2_{11} G(\lambda, \theta)\right]^{-1} \nabla_2 F(\lambda, \theta)$$

Motivation of the framework

Update directions:

$$D_\theta(\theta, v, \lambda) = \nabla_2 G(\lambda, \theta) \quad \text{gradient step toward } \theta^*(\lambda)$$

$$D_v(\theta, v, \lambda) = \nabla^2_{22} G(\lambda, \theta) v + \nabla_2 F(\lambda, \theta)$$

$$\text{gradient step toward } -\left[\nabla^2_{11} G(\lambda, \theta)\right]^{-1} \nabla_2 F(\lambda, \theta)$$

$$D_\lambda(\theta, v, \lambda) = \nabla^2_{12} G(\lambda, \theta) v + \nabla_1 F(\lambda, \theta)$$

$$\text{gradient step toward } \lambda^*$$

Motivation of the framework

Update directions:

$$D_\theta(\theta, v, \lambda) = \nabla_2 G(\lambda, \theta) \quad \text{gradient step toward } \theta^*(\lambda)$$

$$D_v(\theta, v, \lambda) = \nabla^2_{22} G(\lambda, \theta) v + \nabla_2 F(\lambda, \theta)$$

$$\text{gradient step toward } -\left[\nabla^2_{11} G(\lambda, \theta)\right]^{-1} \nabla_2 F(\lambda, \theta)$$

$$D_\lambda(\theta, v, \lambda) = \nabla^2_{12} G(\lambda, \theta) v + \nabla_1 F(\lambda, \theta)$$

$$\text{gradient step toward } \lambda^*$$

Algorithm

For $t = 1 \dots T$:

1. Update $\theta^{t+1} = \theta^t - \rho^t D_\theta^t$
2. Update $v^{t+1} = v^t - \rho^t D_v^t$
3. Update $\lambda^{t+1} = \lambda^t - \gamma^t D_\lambda^t$

Stochastic Update Directions:

$$D_\theta(\theta, v, \lambda) = \frac{1}{n}\sum_{i=1}^{n} \nabla_2 G_i(\lambda, \theta)$$

$$D_v(\theta, v, \lambda) = \frac{1}{n}\sum_{i=1}^{n} \nabla^2_{22} G_i(\lambda, \theta)v + \frac{1}{m}\sum_{j=1}^{m} \nabla_2 F_j(\lambda, \theta)$$

$$D_\lambda(\theta, v, \lambda) = \frac{1}{n}\sum_{i=1}^{n} \nabla^2_{12} G_i(\lambda, \theta)v + \frac{1}{m}\sum_{j=1}^{m} \nabla_1 F_j(\lambda, \theta)$$

$\Rightarrow$ Additive expressions with natural stochastic estimators.

SOBA (StOchastic Bilevel Algorithm) directions

Pick $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ and take

$$
\begin{aligned}
D_\theta^t &= \nabla_2 G_i(\lambda^t, \theta^t) \\
D_v^t &= \nabla_{22}^2 G_i(\lambda^t, \theta^t) v^t + \nabla_2 F_j(\lambda^t, \theta^t) \\
D_\lambda^t &= \nabla_{12}^2 G_i(\lambda^t, \theta^t) v^t + \nabla_1 F_j(\lambda^t, \theta^t)
\end{aligned}
$$

Theoretical guarantees of SOBA

Theorem (Convergence of SOBA)

Under some regularity assumptions on F and G, if h is bounded, then for decreasing step sizes that verify $\rho^t = \alpha t^{-\frac{2}{5}}$ and $\gamma^t = \beta t^{-\frac{3}{5}}$ for some $\alpha, \beta > 0$, the iterates $(\lambda^t)_{1 \leq t \leq T}$ of SOBA verify

$$\inf_{t \leq T} \mathbb{E}[\|\nabla h(\lambda^t)\|^2] = \mathcal{O}(T^{-\frac{1}{2}}) \ .$$

Natural adaptation of single level stochastic algorithms:

SABA: Adaption of SAGA [Defazio et al. 2014]:

The varianced reduced stochastic gradient estimate is:

$$\nabla_2 G(\lambda^t, \theta^t) = \nabla_2 G_i(\lambda^t, \theta^t) - \nabla_2 G_i(\lambda^{t_i}, \theta^{t_i}) + \frac{1}{n}\sum_{k=1}^{n} \nabla_2 G_k(\lambda^{t_k}, \theta^{t_k}))$$

Variance-reduced solvers [Dagréou et al. 2022, 2023]

Natural adaptation of single level stochastic algorithms:

SABA: Adaption of SAGA [Defazio et al. 2014]:

The varianced reduced stochastic gradient estimate is:

$$\nabla_2 G(\lambda^t, \theta^t) = \nabla_2 G_i(\lambda^t, \theta^t) - \nabla_2 G_i(\lambda^{t_i}, \theta^{t_i}) + \frac{1}{n}\sum_{k=1}^{n} \nabla_2 G_k(\lambda^{t_k}, \theta^{t_k}))$$

SRBA: Adaption of SARHA [Nguyen et al. 2017]

The varianced reduced stochastic gradient estimate is:

$$\nabla_2 G(\lambda^t, \theta^t) = \nabla_2 G_i(\lambda^t, \theta^t) - \nabla_2 G_i(\lambda^{t-1}, \theta^{t-1}) + \nabla_2 G(\lambda^{T_t}, \theta^{T_t})$$

Natural adaptation of single level stochastic algorithms:

SABA: Adaption of SAGA [Defazio et al. 2014]:

The varianced reduced stochastic gradient estimate is:

$$\nabla_2 G(\lambda^t, \theta^t) = \nabla_2 G_i(\lambda^t, \theta^t) - \nabla_2 G_i(\lambda^{t_i}, \theta^{t_i}) + \frac{1}{n}\sum_{k=1}^{n} \nabla_2 G_k(\lambda^{t_k}, \theta^{t_k}))$$

SRBA: Adaption of SARHA [Nguyen et al. 2017]

The varianced reduced stochastic gradient estimate is:

$$\nabla_2 G(\lambda^t, \theta^t) = \nabla_2 G_i(\lambda^t, \theta^t) - \nabla_2 G_i(\lambda^{t-1}, \theta^{t-1}) + \nabla_2 G(\lambda^{T_t}, \theta^{T_t})$$

Similar updates for the 5 quantities:

$$\nabla_2 G(\lambda^t, \theta^t), \quad \nabla_2 F(\lambda^t, \theta^t), \quad \nabla_1 F(\lambda^t, \theta^t)$$
$$\nabla^2_{12} G(\lambda^t, \theta^t) v^t, \quad \nabla^2_{22} G(\lambda^t, \theta^t) v^t$$

Theoretical guarantees [Dagréou et al. 2022, 2023]

We denote $N = m + n$

Theorem (Sample complexity of SABA)

Under some regularity assumptions on F and G, with constant and small enough step sizes, SABA achieves an ϵ-stationary point with a sample complexity of $\mathcal{O}(N^{\frac{2}{3}}\epsilon^{-1})$.

Theorem (Sample complexity of SRBA)

Under some regularity assumptions on F and G, with constant and small enough step sizes, SRBA achieves an ϵ-stationary point with a sample complexity of $\mathcal{O}(N^{\frac{1}{2}}\epsilon^{-1})$.

$\Rightarrow$ This matches the sample complexity of single level algorithms.

$\Rightarrow$ We show that SRBA is near-optimal for a class of bilevel problems.

Hyperparameter selection on ℓ^2 regularized logistic regression

Setting:

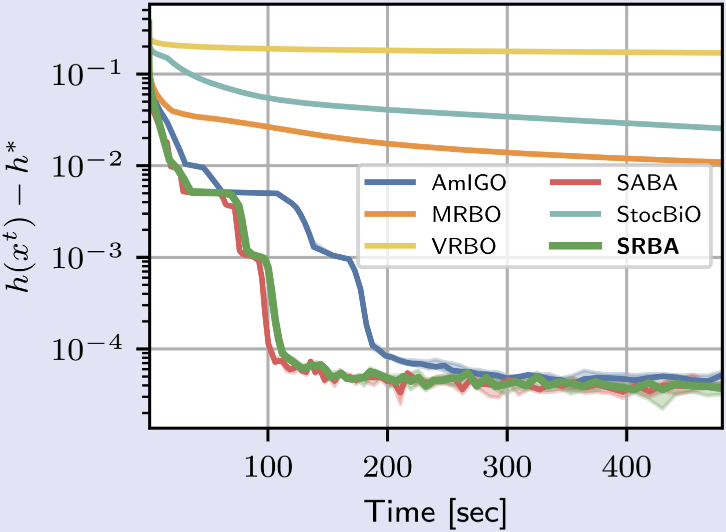
- Task: binary classification
- IJCNN1 dataset: 49 990 training samples, 91 701 validation samples, 22 features
- Training loss:

$$G(\theta, \lambda) = \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i \langle x_i, \theta \rangle)) + \frac{1}{2} \sum_{k=1}^{p} e^{\lambda_k} \theta_k^2$$

- Validation loss: logistic loss

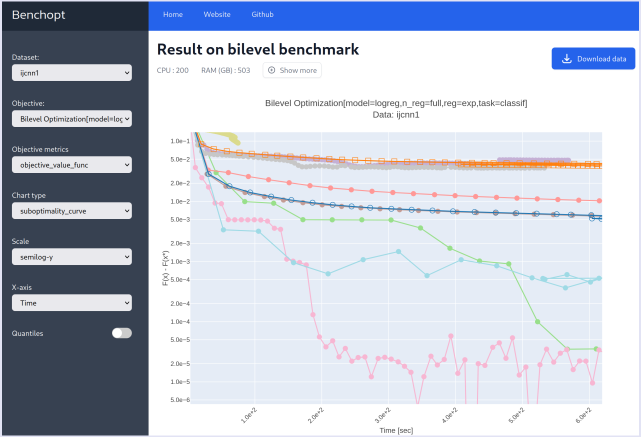
$$F(\theta, \lambda) = \frac{1}{m} \sum_{j=1}^{m} \log(1 + \exp(-y_i^{val} \langle x_i^{val}, \theta \rangle))$$

Hyperparameter selection on ℓ^2 regularized logistic regression



Reproducing this comparison and adding solvers and tasks is easy as:

```
git clone https://github.com/benchopt/benchmark_bilevel
benchopt run ./benchmark_bilevel
```



Benchopt principle

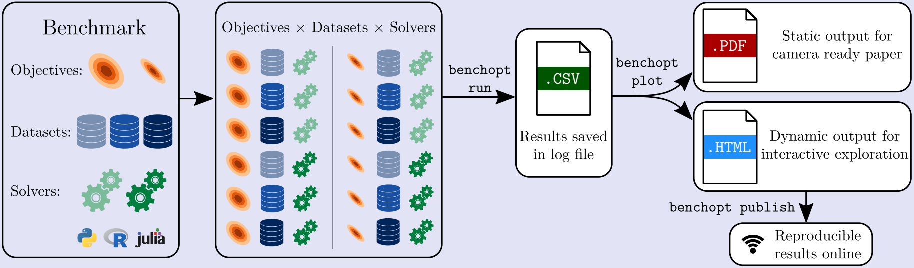
A benchmark is a directory with:

- An `objective.py` file with an `Objective` that define the metrics.
- A directory `datasets` with `Dataset` that define inner and outer tasks.
- A directory `solvers` with one file per `Solver`

```
my_benchmark/
├── README.rst
├── datasets
│   ├── simulated.py  # some dataset
│   └── real.py  # some dataset
├── objective.py  # contains the definition of the objective
└── solvers
    ├── solver1.py  # some solver
    └── solver2.py  # some solver
```

The `benchopt` client runs a cross product and generates a parquet file + HTML visualisation to explore the results.

Benchopt: principle



$\Rightarrow$ Each object can be parametrized so multiple scenario can be tested.

Making tedious tasks easy:

- Sharing code
- Adding methods
- Exploring results
- Varying hyperparameters
- Running in Parallel
- Caching
- ...

Conclusion

- The propose framework allows to adapt many single-level algorithms to the bilevel setting.
- We get similar convergence rate in the bilevel setting, *provided that we solve the inner problem fast enough*.
- `Benchopt` provides a benchmark to quickly test many ideas.
- One limitation is often the selection of learning rates.

$\Rightarrow$ Toward adaptive algorithms for bilevel optimization?

Slides will be on my web page:

tommoral.github.io @tomamoral

Algorithm Unrolling

Differentiable inner problem solvers

References

- Shaban, A., Cheng, C.-A., Hatch, N., and Boots, B. (2019). Truncated Back-propagation for Bilevel Optimization, In *AISTAT*
- Ablin, P., Peyré, G., and **TM** (2020). Super-efficiency of automatic differentiation for functions defined as a minimum, In *ICML*
- Malézieux, B., Michel, F., Kowalski, M., and **TM** (2022). Where prior learning can and can't work in unsupervised inverse problems

Differentiable unrolling of θ^t

Idea: Compute $\frac{\partial \theta^t}{\partial \lambda}(\lambda) \approx \frac{\partial \theta^*}{\partial \lambda}(\lambda)$ using automatic differentiation through an iterative algorithm.

Differentiable unrolling of θ^t

Idea: Compute $\frac{\partial \theta^t}{\partial \lambda}(\lambda) \approx \frac{\partial \theta^*}{\partial \lambda}(\lambda)$ using automatic differentiation through an iterative algorithm.

For the gradient descent algorithm:

$$\theta^{t+1} = \theta^t - \rho \frac{\partial G}{\partial \theta}(\lambda, \theta^t)$$

The Jacobian reads,

$$\frac{\partial \theta^{t+1}}{\partial \lambda}(\lambda) = \left(Id - \rho \frac{\partial^2 G}{\partial \theta^2}(\lambda, \theta^t) \right) \frac{\partial \theta^t}{\partial \lambda}(\lambda) - \rho \frac{\partial^2 G}{\partial \theta \partial \lambda}(\lambda, \theta^t)$$

Differentiable unrolling of θ^t

Idea: Compute $\frac{\partial \theta^t}{\partial \lambda}(\lambda) \approx \frac{\partial \theta^*}{\partial \lambda}(\lambda)$ using automatic differentiation through an iterative algorithm.

For the gradient descent algorithm:

$$\theta^{t+1} = \theta^t - \rho \frac{\partial G}{\partial \theta}(\lambda, \theta^t)$$

The Jacobian reads,

$$\frac{\partial \theta^{t+1}}{\partial \lambda}(\lambda) = \left(Id - \rho \frac{\partial^2 G}{\partial \theta^2}(\lambda, \theta^t) \right) \frac{\partial \theta^t}{\partial \lambda}(\lambda) - \rho \frac{\partial^2 G}{\partial \theta \partial \lambda}(\lambda, \theta^t)$$

$\Rightarrow$ Under smoothness conditions, if θ^t converges to θ^*, this converges toward $\frac{\partial \theta^*}{\partial \lambda}(\lambda)$

Context: min-min problems where $F = G$

$$\Rightarrow \text{Here, } \frac{\partial F}{\partial \theta}(\lambda, \theta^*) = 0$$

Context: min-min problems where $F = G$

$$\Rightarrow \text{Here, } \frac{\partial F}{\partial \theta}(\lambda, \theta^*) = 0$$

We consider the 3 gradient estimates:

- $g_1 = \frac{\partial G}{\partial \lambda}(\lambda, \theta^t)$ Analysis
- $g_2 = \frac{\partial G}{\partial \lambda}(\lambda, \theta^t) + \frac{\partial G}{\partial \theta}(\lambda, \theta^t)\frac{\partial \theta^t}{\partial \lambda}$ Automatic
- $g_3 = \frac{\partial G}{\partial \lambda}(\lambda, \theta^t) - \frac{\partial G}{\partial \theta}(\lambda, \theta^t)\frac{\partial^2 G}{\partial \theta^2}^{-1}(\lambda, \theta^t)\frac{\partial^2 G}{\partial \theta \partial \lambda}(\lambda, \theta^t)$ Implicit

Analysis for min-min problems [Ablin et al. 2020]

Context: min-min problems where $F = G$

$$\Rightarrow \text{Here, } \frac{\partial F}{\partial \theta}(\lambda, \theta^*) = 0$$

We consider the 3 gradient estimates:

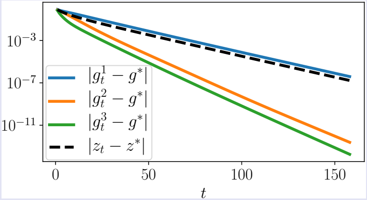
- $g_1 = \frac{\partial G}{\partial \lambda}(\lambda, \theta^t)$ Analysis
- $g_2 = \frac{\partial G}{\partial \lambda}(\lambda, \theta^t) + \frac{\partial G}{\partial \theta}(\lambda, \theta^t)\frac{\partial \theta^t}{\partial \lambda}$ Automatic
- $g_3 = \frac{\partial G}{\partial \lambda}(\lambda, \theta^t) - \frac{\partial G}{\partial \theta}(\lambda, \theta^t)\frac{\partial^2 G}{\partial \theta^2}^{-1}(\lambda, \theta^t)\frac{\partial^2 G}{\partial \theta \partial \lambda}(\lambda, \theta^t)$ Implicit

Convergence rates: For G strongly convex in θ,

$$|g_t^1(x) - g^*(x)| = O\left(|\theta^t(\lambda) - \theta^*(\lambda)|\right),$$

$$|g_t^2(x) - g^*(x)| = o\left(|\theta^t(\lambda) - \theta^*(\lambda)|\right),$$

$$|g_t^3(x) - g^*(x)| = O\left(|\theta^t(\lambda) - \theta^*(\lambda)|^2\right).$$



Analysis for non-smooth min-min problems [Malezieux et al. 2022]

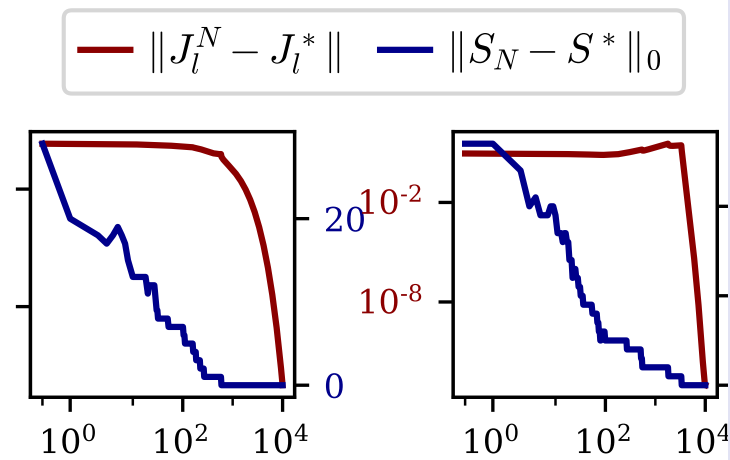
Context: dictionary learning, $F = G$ with an ℓ_1-regularization for θ.

Issue: The implicit gradient quality mostly depends on the support identifiaction,

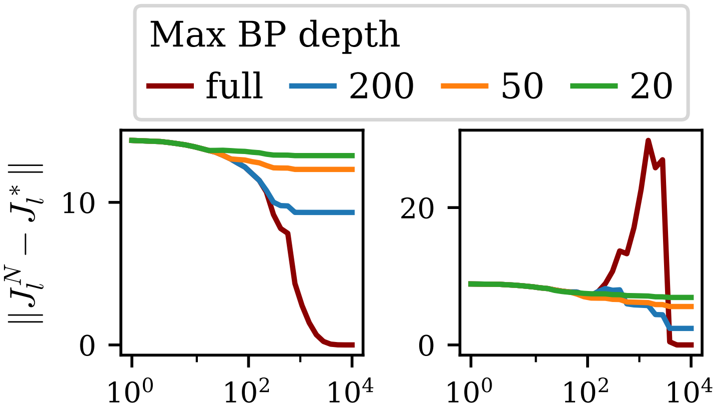
$$\left(\frac{\partial \theta^*}{\partial D_l}\right)_{S^*} = -(D_{:,S^*}^\top D_{:,S^*})^{-1}(D_l \theta^{*\top} + (D_l^\top \theta^* - y_l) Id_n)_{S^*} \ ,$$

$\Rightarrow$ Is the autodiff approach better than the analytic one?

On the support, the function is smooth and we recover the same convergence.



Outside of the support, errors can accumulate and the gradient can blow up.



Hypergradient computation

References

- Lorraine, J., Vicol, P., and Duvenaud, D. (2020). Optimizing millions of hyperparameters by implicit differentiation, In *AISTATS*

Linear system approximation v^*

Solving the linear system for $v^*(\lambda^t)$,

- Core idea is to not inverse the hessian $\frac{\partial^2 G}{\partial \theta^2}(\lambda^t, \theta^t)$,

  We are only interested in one direction.

- Only rely on Hessian-vector product (Hvp).

  Can be computed efficiently

Proposed Methods:

- L-BFGS
- Conjugate Gradient
- Jacobian-Free method

$$\frac{\partial^2 G}{\partial \theta^2}(\lambda^t, \theta^t) \approx Id$$

- Neumann iterations

$$\frac{\partial^2 G}{\partial \theta^2}(\lambda^t, \theta^t)^{-1} \approx \sum_k \left(Id - \frac{\partial^2 G}{\partial \theta^2}(\lambda^t, \theta^t)\right)^k$$

[Pedregosa 2016, Lorraine et al. 2020, Luketina et al. 2016]