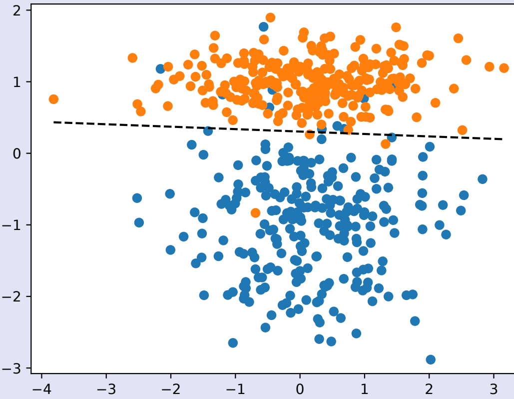
SHINE: Sharing the Inverse Estimate for bi-level optimization.

Thomas Moreau INRIA Saclay

Joint work with Z. Ramzi, S. Bai, F. Mannel, J.-L. Starck & P. Ciuciu

Setup:

- Binary classification task $(X_i, y_i)_{i=1}^{N} \in \mathbb{R}^p \times \{-1, 1\}$
- Linear model: predict y from X with $\text{sign}(\langle \theta, X \rangle)$.



Regularized Logistic loss:

$$G(\lambda, \theta) = \frac{1}{N} \sum_{i=1}^{N} \log(1 + e^{-y_i \langle \theta, X_i \rangle}) + \lambda \|\theta\|_2^2$$

Training the model:

$$\theta^*(\lambda) = \underset{\theta}{\operatorname{argmin}}\, G(\lambda, \theta)$$

Regularized Logistic loss:

$$G(\lambda, \theta) = \frac{1}{N} \sum_{i=1}^{N} \log(1 + e^{-y_i \langle \theta, X_i \rangle}) + \lambda \|\theta\|_2^2$$

Training the model:

$$\theta^*(\lambda) = \underset{\theta}{\operatorname{argmin}}\, G(\lambda, \theta)$$

$\Rightarrow$ **Choose λ using validation data.**

Evaluating the generalization

We want to find λ that ensure the best *generalization* of $\theta^*(\lambda)$.

Validation loss: use held out data $(X_i^{val}, y_i^{val})_{i=1}^M$

$$F(\theta) = \frac{1}{M}\sum_{i=1}^{M} \log(1 + e^{-y_i^{val}\langle\theta, X_i^{val}\rangle})$$

Independent estimate of the risk of the model.

Evaluating the generalization

We want to find λ that ensure the best *generalization* of $\theta^*(\lambda)$.

Validation loss: use held out data $(X_i^{val}, y_i^{val})_{i=1}^M$

$$F(\theta) = \frac{1}{M}\sum_{i=1}^{M} \log(1 + e^{-y_i^{val}\langle\theta, X_i^{val}\rangle})$$

Independent estimate of the risk of the model.

$\Rightarrow$ Find λ that gives a model $\theta^*(\lambda)$ with a good validation loss.

Mathematical rewritting: $\begin{cases} \quad \min_\lambda F(\theta^*(\lambda)) \\ s.t. \quad \theta^*(\lambda) = \operatorname{argmin}_\theta G(\lambda, \theta) \end{cases}$

Bi-level optimization

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

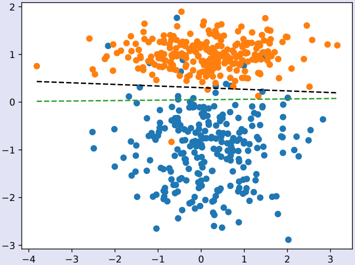
Inner function/Problem

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.

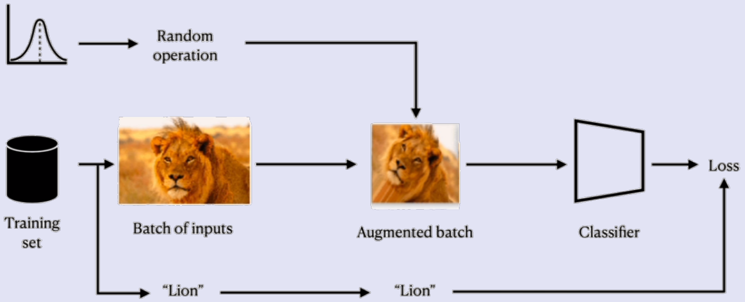
Hyperparameter optimization: λ is a regularization parameter:



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.

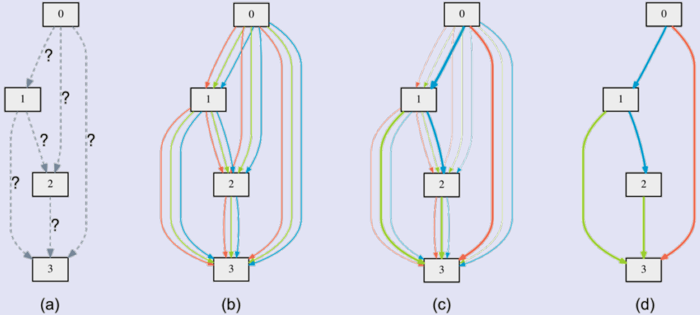
Data augmentation: λ parametrizes the transformations distribution.



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.



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

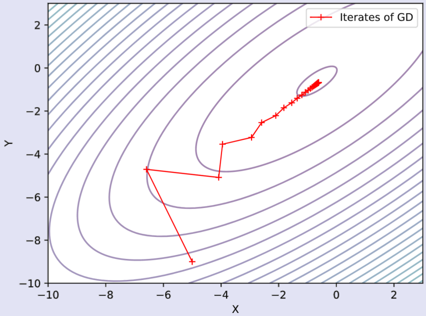
Solving bi-level optimization

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.



Computing the gradient of h

Value function definition:

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

Chain rule:

$$\nabla_\lambda h(\lambda) = \nabla_1 F(\lambda, \theta^*(\lambda)) + (d\theta^*(\lambda))^T \nabla_2 F(\lambda, \theta^*(\lambda))$$

Optimality condition for θ^*

$$\nabla_2 G(\lambda, \theta^*(\lambda)) = 0$$

Jacobian of θ^* - implicit differentiation

Optimality condition for θ^*

$$\nabla_2 G(\lambda, \theta^*(\lambda)) = 0$$

Derivating this equation relative to λ gives:

$$\nabla^2_{22} G(\lambda, \theta^*(\lambda)) d\theta^*(\lambda) + \nabla^2_{21} G(\lambda, \theta^*(\lambda)) = 0,$$

Jacobian of θ^* - implicit differentiation

Optimality condition for θ^*

$$\nabla_2 G(\lambda, \theta^*(\lambda)) = 0$$

Derivating this equation relative to λ gives:

$$\nabla^2_{22} G(\lambda, \theta^*(\lambda)) d\theta^*(\lambda) + \nabla^2_{21} G(\lambda, \theta^*(\lambda)) = 0,$$

Implicit function theorem

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

Computing the gradient of h

Value function gradient:

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

Computing the gradient of h

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 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 bi-level optimization

References

- Pedregosa, F. (2016). Hyperparameter optimization with approximate gradient. In *International Conference on Machine Learning (ICML)*, pages 737–746, New-York, NY, USA

Do we need to compute θ^* and v^* precisely?

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

Hyperparameter optimization with Approximate Gradient HOAG [Pedregosa 2016]

Do we need to compute θ^* and v^* precisely?

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

- Compute θ^t such that $\|\theta^t - \theta^*(\lambda^t)\|_2 \leq \epsilon_t$,
  iterative solver *e.g.* L-BFGS
- Compute v^t such that $\|\frac{\partial^2 G}{\partial \theta^2}(\lambda^t, \theta^t) v^t + \frac{\partial F}{\partial \theta}(\lambda^t, \theta^t)\|_2 \leq \epsilon_t$,
  L-BFGS or CG
- Compute the approximate gradient $g_t = \frac{\partial F}{\partial \lambda}(\lambda^t, \theta^t) + \frac{\partial^2 G}{\partial \theta \partial \lambda}(\lambda^t, \theta^t) v^t$
- Update the outer variable $\lambda^{t+1} = \lambda^t - \rho^t g^t$

Theorem: If $\sum_t \epsilon_t < \infty$ and the step-sizes are chosen appropriatly, then the algorithm converges to a stationary point *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:

- L-BFGS
- Jacobian-Free method

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

- Algorithm unrolling
- Conjugate Gradient
- Neumann iterations

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

- Use Quasi-newton hessian approximation

[Pedregosa 2016, Lorraine et al. 2020, Luketina et al. 2016, Ramzi et al. 2022]

SHINE - Sharing the INverse Estimate

References

- Ramzi, Z., Mannel, F., Bai, S., Starck, J.-L., Ciuciu, P., and Moreau, T. (2022). SHINE: SHaring the INverse Estimate from the forward pass for bi-level optimization and implicit models. In *International Conference on Learning Representations (ICLR)*, online

Quasi Newton 101:

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

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

Inverse with Sherman-Morrison

Quasi Newton 101:

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

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

Inverse with Sherman-Morrison

$\Rightarrow$ The Hessian for v^* is the same as the one from the inner problem.

Idea: reuse the approximation of the Hessian B_t computed by L-BFGS for the inner problem.

$$\begin{cases} \tilde{v}_t = B_t^{-1} \nabla_2 F(\lambda, \theta^t) \\ \tilde{\nabla} h(\lambda) = \nabla_1 F(\lambda, \theta^t) + \nabla^2_{12} G(\lambda, \theta^t) \tilde{v}_t \end{cases}$$

Properties of B:

Idea: reuse the approximation of the Hessian B_t computed by L-BFGS for the inner problem.

$$\begin{cases} \tilde{v}_t = B_t^{-1} \nabla_2 F(\lambda, \theta^t) \\ \tilde{\nabla} h(\lambda) = \nabla_1 F(\lambda, \theta^t) + \nabla_{12}^2 G(\lambda, \theta^t) \tilde{v}_t \end{cases}$$

Properties of B:

- It is computed when solving $\theta^* = \operatorname{argmin}_\theta G(\theta)$ using a quasi-Newton method.

Idea: reuse the approximation of the Hessian B_t computed by L-BFGS for the inner problem.

$$\begin{cases} \tilde{v}_t = B_t^{-1} \nabla_2 F(\lambda, \theta^t) \\ \tilde{\nabla} h(\lambda) = \nabla_1 F(\lambda, \theta^t) + \nabla^2_{12} G(\lambda, \theta^t) \tilde{v}_t \end{cases}$$

Properties of B:

- It is computed when solving $\theta^* = \operatorname{argmin}_\theta G(\theta)$ using a quasi-Newton method.
- It is easily invertible using the Sherman-Morrison formula, because low-rank.

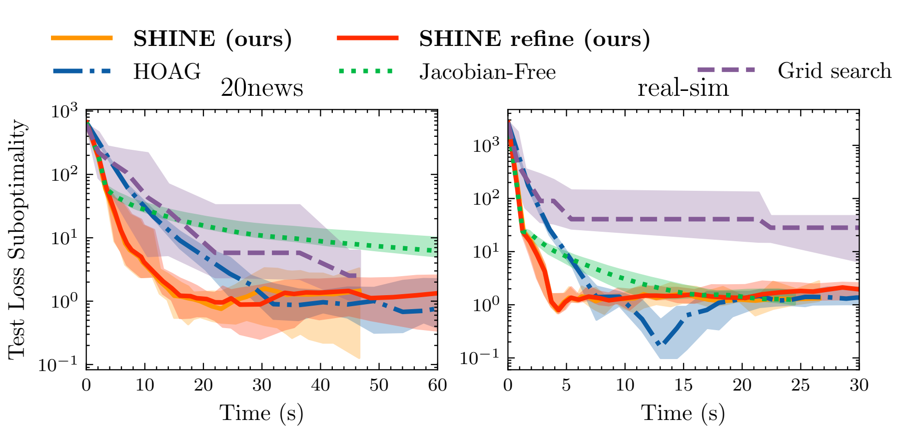
SHINE direction convergence

Theorem (Convergence of SHINE to the Hypergradient using ULI)

Under the Uniform Linear Independence (ULI) assumption and some additional smoothness and convexity assumptions, for a given parameter λ*,* (θ^t) *converges q-superlinearly to* $\theta^\star$ *and*

$$\lim_{t \to \infty} \nabla_1 F(\lambda, \theta^t) + \nabla^2_{12} G(\lambda, \theta^t) \tilde{v}_t = \nabla h(\lambda).$$

Logistic Regression with ℓ_2-regularisation on 2 datasets:



Application to Deep Equilibrium Networks (DEQs)

A recurrent expression of classical, explicit networks:

$$\theta_n = f_{\lambda_n}(\theta_{n-1}), \quad \forall n < N$$

A recurrent expression of classical, explicit networks:

$$\theta_n = f_{\lambda_n}(\theta_{n-1}), \quad \forall n < N$$

What if $N \to \infty$?
If we suppose $\lambda_n = \lambda, \forall n$:

$$\theta^\star = f_\lambda(\theta^\star)$$

Deep Equilibrium networks

Deep Equilibrium networks (DEQs) [Bai et al., 2019] are a type of implicit model. The output is the solution to a fixed-point equation.

$$h_\lambda(x) = \theta^\star, \text{ where } \theta^\star = f_\lambda(\theta^\star, x)$$

Deep Equilibrium networks (DEQs) [Bai et al., 2019] are a type of implicit model. The output is the solution to a fixed-point equation.

$$h_\lambda(x) = \theta^\star, \text{ where } \theta^\star = f_\lambda(\theta^\star, x)$$

This approximates an infinite depth network:

$$\theta_n = f_\lambda(\theta_{n-1}), \quad \forall n \to \infty$$

In practice, we work with root finding algorithms using $g_\lambda = id - f_\lambda$.

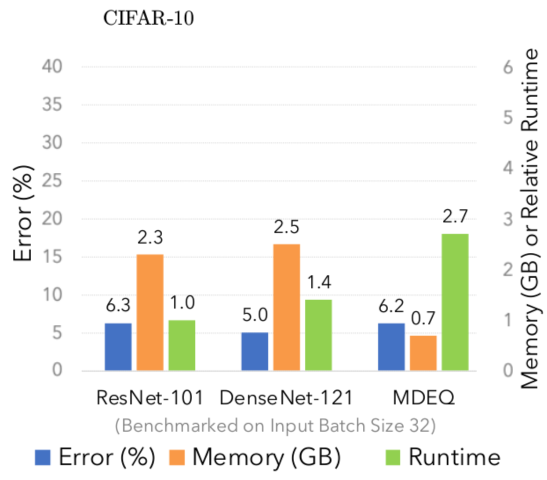
Deep Equilibrium Network:

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

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

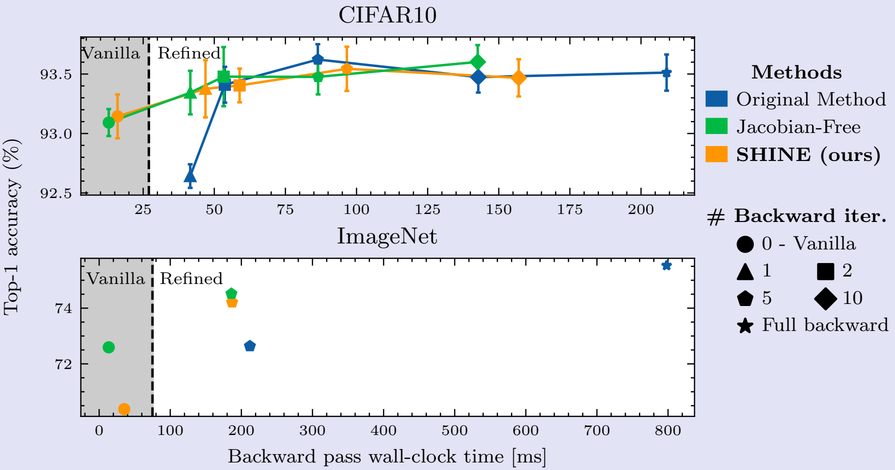
In pratice, use quasi-Newton algorithm such as the Broyden method for both finding θ^* and its derivative v^*.

Deep Equilibrium Network



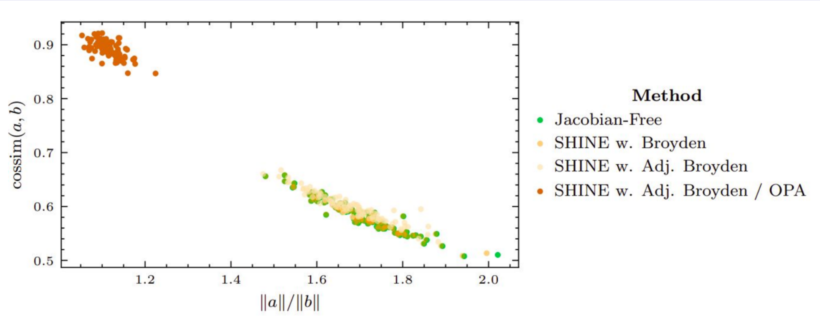
- **Efficient memory**
- **Slow runtime**

Multiscale DEQ on CIFAR10:



B^{-1} is not a uniformly good approximation.

OPA: add additional secant conditions for B update.



$\Rightarrow$ Better gradient approximation (theoretical and empirical)

However, this does not improve results for test error.

Conclusion

- Bi-level optimization is intrinsic in many ML problems.
- We propose to re-use by-product from θ^* computation to make it easy to get v^*.
- Good results for HO but still open questions for DEQs.

Z. Ramzi S. Bai F. Mannel J.L. Starck P. Ciuciu

Slides will be on my web page:

tommoral.github.io @tomamoral