Stabilizing Bi-Level Hyperparameter Optimization using Moreau-Yosida Regularization

Sauptik Dhar   SAUPTIK.DHAR@LGE.COM
America Research Lab, LG Electronics

Unmesh Kurup   UNMESH.KURUP@LGE.COM
America Research Lab, LG Electronics

Mohak Shah   MOHAK.SHAH@LGE.COM
America Research Lab, LG Electronics

Abstract

This research proposes to use the Moreau-Yosida envelope to stabilize the convergence behavior of bi-level Hyperparameter optimization solvers, and introduces the new algorithm called Moreau-Yosida regularized Hyperparameter Optimization (MY-HPO) algorithm. Theoretical analysis on the correctness of the MY-HPO solution and initial convergence analysis is also provided. Our empirical results show significant improvement in loss values for a fixed computation budget, compared to the state-of-art bi-level HPO solvers.

1. Introduction

Successful application of any machine learning (ML) algorithm heavily depends on the careful tuning of its hyper-parameters. However, hyper-parameter optimization (HPO) is a non-trivial problem and has been a topic of research for several decades. Most existing works broadly adopt one of the following two approaches, a) Black-Box optimization or, b) Direct optimization. The Black-Box optimization approach is agnostic of the underlying ML model and adopts an advanced search algorithm to select the hyper-parameters that minimizes the validation error. Popular examples include, Random search, Grid search, Bayesian Optimization based approaches (Bergstra et al., 2013; Snoek et al., 2012) or other advanced exploration techniques like (Li et al., 2017) etc. The advantage is that it is applicable to any machine learning problem. However, it does not utilize the structure of the underlying ML model, which can be exploited for faster solutions. The Direct Optimization utilizes the underlying ML algorithm structure by parameterizing the validation error as a function of the hyper-parameters, and directly optimizes it. Typical examples include, bi-level hyper-parameter optimization (Lorraine and Duvenaud, 2018; MacKay et al., 2019; Pedregosa, 2016; Franceschi et al., 2017; Maclaurin et al., 2015; Franceschi et al., 2018; Mehra and Hamm, 2019) and bound based analytic model selection (Chapelle et al., 2002; Dhar et al., 2019; Cortes et al., 2017) etc. Such approaches can provide significant computation improvements for HPO. However the limitations of these approaches include, a) the gradients of the validation error w.r.t the hyperparameters must exist and b) instability of the optimization problem under ill-conditioned settings. In this paper we address this unstable convergence behavior of the bi-level HPO approaches for ill-conditioned problems, and propose a new framework to improve it. The main contributions of this work are,

1. We propose a new algorithm called Moreau-Yosida regularized Hyperparameter Optimization (MY-HPO) to stabilize bi-level HPO for ill-conditioned problems.

2. We provide theoretical analysis on the correctness of the proposed MY-HPO algorithm and also provide some initial convergence analysis.

3. Finally, we provide extensive empirical results in support of our proposed approach.

The paper is structured as follows. Section 2 introduces the bi-level formulation for HPO and discusses the existing issues of the state-of-art bi-level HPO solvers (Lorraine and Duvenaud, 2018; MacKay et al., 2019) under ill-conditioned settings. Section 3 introduces the Moreau-Yosida regularized Hyperparameter Optimization (MY-HPO) algorithm as a solution to stabilize bi-level HPO for such settings. Additional theoretical analysis of its solution's correctness and convergence behavior is also provided. Empirical results and discussions are provided in section 4.

2. Bi-Level Hyperparameter Optimization (HPO)

The standard bi-level formulation for hyperparameter optimization (HPO) is given as,

$$\boldsymbol{\lambda}^* \in \underset{\boldsymbol{\lambda}}{\operatorname{argmin}}\, L_V(\boldsymbol{\lambda}, \underset{\mathbf{w}}{\operatorname{argmin}}\, L_T(\mathbf{w}, \boldsymbol{\lambda})) \tag{1}$$

Here, L_V = Validation loss, L_T = Training loss, $\boldsymbol{\lambda}$ = Hyperparameters, $\mathbf{w}$ = Model parameters. A popular approach involves introducing a **best-response function** (Lorraine and Duvenaud, 2018; MacKay et al., 2019) to rather solve,

$$\boldsymbol{\lambda}^* \in \underset{\boldsymbol{\lambda}}{\operatorname{argmin}}\, L_V(\boldsymbol{\lambda}, G_{\boldsymbol{\phi}}(\boldsymbol{\lambda})) \quad \text{s.t.} \quad G_{\boldsymbol{\phi}}(\boldsymbol{\lambda}) \in \underset{\mathbf{w}}{\operatorname{argmin}}\, L_T(\mathbf{w}, \boldsymbol{\lambda}) \tag{2}$$

$G_{\boldsymbol{\phi}}(\boldsymbol{\lambda})$ is the best-response function parameterized by the hyperparameters $\boldsymbol{\lambda}$. For simplicity we limit our discussion to only scalar (single) λ. Of course, it can be easily extended to multiple hyperparameters. The work on Stochastic Hyperparameter Optimization (SHO) (Lorraine and Duvenaud, 2018) use a hypernetwork to model the best-response function as $G_{\boldsymbol{\phi}}(\lambda) = \lambda\phi_1 + \phi_0 = \Lambda\,\boldsymbol{\phi}$, where $\boldsymbol{\phi} = \begin{bmatrix} \phi_1 \\ \phi_0 \end{bmatrix}$ and $\Lambda = [\lambda\mathbf{I}|\mathbf{I}]$; with $\boldsymbol{\phi} \in \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, L_T(\Lambda\,\boldsymbol{\theta}, \lambda)$; and adopts an alternate minimization of the training loss L_T w.r.t $\boldsymbol{\phi}$ (hypernetwork parameters), and validation loss $Ł_V$ w.r.t λ (hyperparameters) to solve the HPO problem. Their proposed algorithm is provided in Appendix B. MacKay et al. (2019) adopts a similar alternating gradient approach but modifies the algorithm hypothesis class (by scaling and shifting the network hidden layers) and adds additional stability through adaptively tuning the degree of stochasticity (i.e. perturbation scale) for the gradient updates. In this work we adopt the SHO algorithm 2 as a representative for such alternating gradients approaches for solving the bi-level HPO.

One major limitation of these alternating gradient based bilevel HPO algorithms is that, the convergence behavior heavily depends on the conditioning of the problem. Basically, for ill-conditioned settings, the step-size used for gradient updates need to be sufficiently small to ensure stability of such algorithms. This in turn leads to poor convergence rates (also see in our results section 4). To alleviate this instability and ensure improved convergence we introduce our Moreau-Yosida regularized Hyperparameter Optimization (MY-HPO) algorithm. For simplicity we will assume unique solutions for (2) in the rest of the paper.

3. Moreau-Yosida regularized HPO

First we reformulate the problem in (2). Given, $\phi^* \in \underset{\boldsymbol{\theta}}{\text{argmin}}\, L_T(\Lambda^*\, \boldsymbol{\theta}, \lambda^*)$ we solve,

$$\min_{\lambda, \mathbf{w}} \quad L_T(\mathbf{w}, \lambda^*) \; + \; L_V(\lambda, G_{\phi^*}(\lambda)) \tag{3}$$

$$\text{s.t.} \quad \mathbf{w} = G_{\phi^*}(\lambda) = \lambda \phi_1^* + \phi_0^* = \Lambda\, \phi^* \;; \qquad \text{where} \quad \phi^* = \begin{bmatrix} \phi_1^* \\ \phi_0^* \end{bmatrix} \quad \text{and } \Lambda = [\lambda \mathbf{I} | \mathbf{I}]$$

Here, we assume that λ^* and ϕ^* are provided to us by an oracle satisfying $\nabla_\phi L_T(\Lambda^*\, \phi, \lambda^*) = 0$ at solution $\Lambda^* = [\lambda^* \mathbf{I} | \mathbf{I}]$. Proposition 1 justifies solving (3) in lieue of (2).

Proposition 1 *For a bijective mapping* $\mathbf{w} = G_{\phi^*}(\lambda)$, *the stationary points* $(\mathbf{w}^*, \lambda^*)$ *of* (3) *are also stationary points of* (2) *with* $\nabla_\mathbf{w} L_T(\mathbf{w}^*, \lambda^*) = 0$ *and* $\nabla_\lambda L_V(\lambda^*, G_{\phi^*}(\lambda^*)) = 0$.

The problem (3) is a sum of two functions parameterized by different arguments connected through an equality constraint. Such formulations are frequently seen in machine learning problems and popularly solved through variants of Alternating Direction Method of Multipliers (Boyd et al., 2011; Goldstein et al., 2014), Alternating Minimization Algorithm (Tseng, 1991) or Douglas Rachford Splitting (Eckstein and Bertsekas, 1992) etc. However, a major difference in (3) is that the oracle solution λ^* and $\phi^* \in \underset{\boldsymbol{\theta}}{\text{argmin}}\, L_T(\Lambda^*\, \boldsymbol{\theta}, \lambda^*) \Rightarrow \nabla_{\boldsymbol{\theta}} L_T(\Lambda^*\, \phi^*, \lambda^*) = 0$ is not available a priori. Hence, we cannot directly apply these existing approaches to solve (3). This leads to our new algorithm called Moreau-Yosida regularized Hyperparameter Optimization (MY-HPO) to solve (3). At iteration $k + 1$ we take the steps,

Step 1. Update $\phi^{k+1} = [\phi_1^{k+1} | \phi_0^{k+1}]$

$$\mathbf{v}^{k+1} = \underset{\mathbf{v}}{\text{argmin}}\, L_T(\mathbf{v}, \lambda^k); \quad \phi_0^{k+1} = \overline{\mathbf{v}^{k+1}} = \sum_j v_j^{k+1}; \quad \phi_1^{k+1} = \frac{\mathbf{v}^{k+1} - \overline{\mathbf{v}^{k+1}}}{\lambda^k} \tag{4}$$

Step 2. $\mathbf{w}^{k+1} = \underset{\mathbf{w}}{\text{argmin}}\, L_T(\mathbf{w}, \lambda^k) + (\mathbf{u}^k)^T(\mathbf{w} - \Lambda^k \phi^{k+1}) + \frac{\rho}{2}||\mathbf{w} - \Lambda^k \phi^{k+1}||_2^2 \tag{5}$

Step 3. $\lambda^{k+1} = \underset{\lambda}{\text{argmin}}\, L_V(\lambda, G_{\phi^{k+1}}(\lambda)) + (\mathbf{u}^k)^T(\mathbf{w}^{k+1} - \Lambda \phi^{k+1}) + \frac{\rho}{2}||\mathbf{w}^{k+1} - \Lambda \phi^{k+1}||_2^2 \tag{6}$

Step 4. Update consensus $\mathbf{u}^{k+1} = \mathbf{u}^k + \rho(\mathbf{w}^{k+1} - \Lambda^{k+1} \phi^{k+1}) \tag{7}$

The complete algorithm is provided in Appendix C (Algorithm 3). Step 1, ensures a unique solution for given λ^k i.e. $\phi^{k+1} = \underset{\boldsymbol{\theta}}{\text{argmin}}\, L_T(\Lambda^k\, \boldsymbol{\theta}, \lambda^k)$. In Steps 2 and 3 rather than taking the gradient updates of L_T, L_V (as in SHO); we take the gradient of the Moreau-Yosida (MY) regularized functions. The Moreau-Yosida (MY) regularization of a function is defined as $f_{1/\rho}(\cdot) := \underset{x}{\min}\, f(x) + \frac{\rho}{2}||\cdot - x||_2$ and serves as a smooth approximate for $f(\cdot)$. Note that (5) and (6) transforms to gradient updates of the MY regularized L_T and L_V in its scaled form (Boyd et al., 2011). This lends to better stability of these updates in Steps $2 - 3$; and is highly desirable for ill-conditioned problems. Another aspect of this algorithm is that now $\mathbf{w}$ and λ updates are not agnostic of each other. The $\mathbf{w} - \Lambda\phi$ terms constrains against larger steps in a direction detrimental to either of the loss functions L_T, L_V. In

---

Algorithm 1: Simplified MY-HPO Algorithm

Input: MAXITERS = Total iterations, ε_{TOL} = convergence error tolerance.
      α, β, δ = step size for gradient updates.

Output: $\lambda, \mathbf{w}$

Initialize: $\lambda \leftarrow -1$ and $\mathbf{v} \leftarrow \mathbf{0}, \mathbf{w} \leftarrow \mathbf{0}, \mathbf{u} \leftarrow \mathbf{0}$

while ($k + 1 \leq$ MAXITERS) and ($convergence\_error \geq \varepsilon_{TOL}$) **do**

$\quad \mathbf{v}^{k+1} = \mathbf{v}^k - \alpha \nabla_{\mathbf{v}} L_T; \ \boldsymbol{\phi}_0^{k+1} = \overline{\mathbf{v}^{k+1}}; \ \boldsymbol{\phi}_1^{k+1} = \dfrac{\mathbf{v}^{k+1} - \overline{\mathbf{v}^{k+1}}}{\lambda}$ and $\boldsymbol{\phi}^{k+1} = [\boldsymbol{\phi}_1^{k+1} | \boldsymbol{\phi}_0^{k+1}]$

$\quad \mathbf{w}^{k+1} = \mathbf{w}^k - \beta[\nabla_{\mathbf{v}} L_T + \mathbf{u}^k + \rho(\mathbf{w}^k - \Lambda^k \boldsymbol{\phi}^{k+1})]$

$\quad \lambda^{k+1} = \lambda^k - \delta \nabla_\lambda [L_V(\lambda, G_\phi(\lambda)) + (\mathbf{u}^k)^T(\mathbf{w}^{k+1} - \Lambda \boldsymbol{\phi}^{k+1}) + \dfrac{\rho}{2}||\mathbf{w}^{k+1} - \Lambda \boldsymbol{\phi}^{k+1}||_2^2]$

$\quad \mathbf{u}^{k+1} = \mathbf{u}^k + \rho(\mathbf{w}^{k+1} - \Lambda^{k+1} \boldsymbol{\phi}^{k+1})$

end

---

essence these additional terms maintain consensus between the updates while minimizing L_T w.r.t $\mathbf{w}$, and L_V w.r.t λ separately. Any deviation from this consensus is captured in $\mathbf{u}$-update and fed back in the next iteration. The user-defined, $\rho \geq 0$ controls the scale of these augmented terms and hence convergence of the algorithm. The proposed algorithm has the following convergence properties,

Proposition 2 (Convergence Criteria) The necessary and sufficient conditions for Steps $1-4$ to solve (3) are, $r^{k+1} = \mathbf{w}^{k+1} - \Lambda^{k+1} \boldsymbol{\phi}^{k+1} \to 0$ and $s^{k+1} = \rho(\Lambda^{k+1} \boldsymbol{\phi}^{k+1} - \Lambda^k \boldsymbol{\phi}^{k+1}) \to 0$.

Further we make the following claim on the algorithm's convergence to its stationary points.

Claim 1 (Convergence Guarantee) Under the assumptions that L_T, L_V are proper, closed and convex; with L_T strongly convex and $\{\boldsymbol{\phi}^k\}$ is bounded $\forall k$. The steps $1 - 4$ converges to a stationary point. Further at stationary point we have $r^{k+1} = 0$ and $s^{k+1} = 0$.

All proofs are provided in Appendix. To maintain same per-step iteration cost as the SHO algorithm we simplify the steps 1-4 further and rather take single gradient updates in Steps 1-3. This results to the simplified MY-HPO Algorithm 1 which is used throughout our experiments. For simplicity we call this simplified Algorithm 1 as MY-HPO through out the paper. Additionally, to ensure descent direction we also add a backtracking (BT) of step-size scaled by a factor of 0.5 (see chapter 4 in (Beck, 2014)). The refer to the algorithm with backtracking as MY-HPO(BT) and without it i.e. constant step-size as MY-HPO (C).

4. Results and Summary

4.1 Experimental Settings

We provide analysis for both regression and classification problems using the loss functions,
- Least Square: $L_V = \frac{1}{2N_V} \sum_{\mathbf{x}_i \in \mathcal{V}}(y_i - \mathbf{w}^T\mathbf{x}_i)^2$, $L_T = \frac{1}{2N_T} \sum_{\mathbf{x}_i \in \mathcal{T}}(y_i - \mathbf{w}^T\mathbf{x}_i)^2 + e^\lambda ||\mathbf{w}||_2^2$
- Logistic: $L_V = \frac{1}{N_V} \sum_{\mathbf{x}_i \in \mathcal{V}} \log(1 + e^{-y_i \mathbf{w}^T\mathbf{x}_i})$, $L_T = \frac{1}{N_T} \sum_{\mathbf{x}_i \in \mathcal{T}} \log(1 + e^{-y_i \mathbf{w}^T\mathbf{x}_i}) + e^\lambda ||\mathbf{w}||_2^2$

Table 1: Datasets and Experimental Settings

Data Sets	Training/Validation size	Test size	Dimension (d)
COOKIE (Osborne et al., 1984)	34/17 (50%/25%)	19 (25%)	699 (NIR)
MNIST Regression (LeCun, 1998)	1000 (100 per digit)	5000 (500 per digit)	784 (Pixel)
MNIST Classification (LeCun, 1998)	500 (250 per class)	1000 (500 per class)	784 (Pixel)
GTSRB (Stallkamp et al., 2012)	1000 (500 per class)	1000 (500 per class)	1568 (HOG)

Here, $\mathbf{x}_i \in \Re^d$, and $\mathbf{y} \in \Re$ (regression) or $\{-1, 1\}$ (classification), $d =$ dimension of the problem, $\mathcal{T} =$ Training data, $N_T =$ number of training samples and $\mathcal{V} =$ Validation Data, $N_V =$ number of validation samples. The experimental details are provided in Table 1. Here, for regression, we use the MNIST data to regress on the digit labels '0' - '9' (Park et al., 2020) and the COOKIE to predict the percentage of 'fat' w.r.t the 'flour' content using near infrared region (NIR) values (Osborne et al., 1984). For classification, we classify between digits '0' vs. '1' for MNIST, and the traffic signs '30' vs. '80' for GTSRB.
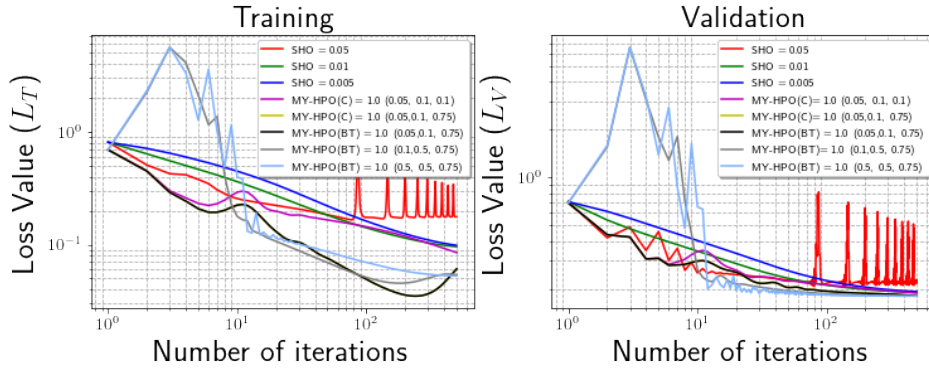
4.2 Results

Table 2 provides the average ($\pm$ standard deviation) of the loss values on training, validation and a separate test data, over 10 runs of the experiments. Here, for each experiment we partition the data sets in the same proportion as shown in Table 1. For regression we scale the output's loss with the variance of y $-$ values. This is a standard normalization technique which illustrates the proportion of un-explained variance by the estimated model. For the SHO and MY-HPO algorithms we only report the performance of the models using the best performing step-sizes. A detailed ablation study using different step-sizes for the algorithms is provided in Appendix D. In addition we also provide the results for popular black-box algorithms publicly available through the Auptimizer toolbox (Liu et al., 2019). Here for all the algorithms we maintain a fixed budget of gradient computations. Additional details on the various algorithm's parameter settings are provided in Appendix D.

Table 2 shows that for both the classification and regression problems MY-HPO algorithm significantly outperforms all the baseline algorithms. The dominant convergence behavior of the bi-level formulations (i.e. MY-HPO and SHO) compared to black-box approaches is well known from previous studies (Lorraine and Duvenaud, 2018; MacKay et al., 2019). This is also seen in our results. Of course, increasing the number of gradient computations allows the black-box approaches achieve similar loss values (see Appendix D). However, a non-trivial observation is that MY-HPO significantly outperforms the SHO algorithm. To further analyze this improved convergence behavior of the MY-HPO algorithm; we provide the convergence curves of these algorithms for Logistic loss using GTSRB data in Fig 1. Here, MY-HPO (BT) involves back-tracking to ensure descent direction; whereas MY-HPO (C) uses constant step updates. Fig 1 shows that the SHO algorithm obtains best results with the step-size fixed to $\alpha = 0.01$. Increasing it further to $\alpha = 0.05$ destabilizes the updates and results to sub-optimal SHO solution. On the other hand, using the Moreau Yosida regularized updates we can accomodate for larger step-sizes and hence achieve better convergence rates. Additional improvement can be expected by using back-tracking as it ensures descent directions which adds to the stability of the algorithm. This behavior is persistently seen for both the classification and regression problems for all the data sets (see Appendix D).

Table 2: Loss values of different HPO algorithms using fixed computation budget.

DATA	SHO	MYHPO (C)	MYHPO (BT)	RANDOM	GRID	HYPEROPT	SPEARMINT
Regression Problems							
Cookie		No. of gradient computations = 5000					
TRAIN $(\times 10^{-2})$	71.5 ± 66.9	7.7 ± 2.7	$\mathbf{5.4 \pm 2.5}$	89.1 ± 10.3	6.7 ± 2.2	58.1 ± 7.9	6.7 ± 2.2
VAL. $(\times 10^{-2})$	63.1 ± 72.8	17.6 ± 13.4	$\mathbf{6.2 \pm 4.8}$	93.4 ± 26.4	16.7 ± 12.9	49.7 ± 19.5	16.7 ± 12.9
TEST $(\times 10^{-2})$	75.3 ± 98.7	17.4 ± 15.8	$\mathbf{6.9 \pm 7.3}$	85.6 ± 28.9	16.1 ± 14.7	47.2 ± 23.5	16.1 ± 14.7
MNIST		No. of gradient computations = 6000					
TRAIN $(\times 10^{-2})$	26.7 ± 2.7	$\mathbf{22.6 \pm 0.6}$	$\mathbf{22.6 \pm 0.6}$	22.3 ± 0.6	17.4 ± 0.5	22.8 ± 5.7	17.4 ± 0.5
VAL. $(\times 10^{-2})$	24.6 ± 1.4	$\mathbf{23.7 \pm 1.0}$	$\mathbf{23.7 \pm 1.0}$	25.6 ± 2.8	26.3 ± 2.4	25.0 ± 2.5	26.3 ± 2.5
TEST $(\times 10^{-2})$	23.4 ± 1.1	$\mathbf{22.3 \pm 0.55}$	$\mathbf{22.3 \pm 0.55}$	24.6 ± 2.8	24.8 ± 2.2	23.8 ± 2.1	24.8 ± 2.2
Classification Problems							
MNIST		No. of gradient computations = 1000					
TRAIN $(\times 10^{-2})$	5.58 ± 2.1	2.94 ± 1.6	$\mathbf{2.87 \pm 1.6}$	35.7 ± 71.4	0.3 ± 0.1	5.18 ± 0.35	0.3 ± 0.1
VAL. $(\times 10^{-2})$	4.9 ± 1.96	4.4 ± 1.89	$\mathbf{4.3 \pm 1.91}$	19.1 ± 37.8	5.31 ± 2.7	4.9 ± 1.46	5.31 ± 2.7
TEST $(\times 10^{-2})$	5.32 ± 1.6	5.1 ± 1.84	$\mathbf{5.0 \pm 1.79}$	19.4 ± 38.6	6.22 ± 2.8	5.19 ± 1.44	6.22 ± 2.8
GTSRB		No. of gradient computations = 1000					
TRAIN $(\times 10^{-2})$	6.83 ± 1.04	3.26 ± 2.5	$\mathbf{4.31 \pm 1.2}$	47.48 ± 117.3	0.09 ± 0.01	11.9 ± 30.5	0.09 ± 0.01
VAL. $(\times 10^{-2})$	14.35 ± 2.34	14.35 ± 2.38	$\mathbf{13.98 \pm 2.3}$	39.85 ± 75.4	23.5 ± 6.6	18.9 ± 11.7	23.5 ± 6.6
TEST $(\times 10^{-2})$	14.04 ± 1.82	13.8 ± 2.25	$\mathbf{13.59 \pm 1.9}$	38.6 ± 70.6	21.8 ± 5.2	17.6 ± 8.5	21.8 ± 5.2

Figure 1: Convergence behavior of SHO vs. MY-HPO for different step-sizes for GTSRB data using Logistic loss. For MY-HPO we show the $\rho(\alpha, \beta, \delta)$ values.

4.3 Summary

In summary our results confirm the effectiveness of Moreau-Yosida (MY) regularized updates for bi-level HPO under (ill-conditioned) limited data settings. Here rather than taking alternating gradient updates as in SHO (Lorraine and Duvenaud, 2018) or STN (MacKay et al., 2019); we propose a modified algorithm MY-HPO by taking gradient updates on the Moreau-Yosida envelope and maintaining a consensus variable. The proposed MY-HPO algorithm provides added stability and enables us to use larger step sizes which inturn leads to better convergence rates. Under a fixed computation budget MY-HPO significantly outperforms the SHO algorithm (a popular representative for alternating gradient based bi-level HPO solvers); and the widely used black-box HPO routines. Owing to space constraints details regarding the algorithm parameters for reproducing the experimental results are provided in Appendix D.

References

Amir Beck. *Introduction to nonlinear optimization: Theory, algorithms, and applications with MATLAB*, volume 19. Siam, 2014.

James Bergstra, Daniel Yamins, and David Daniel Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. 2013.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

Olivier Chapelle, Vladimir Vapnik, Olivier Bousquet, and Sayan Mukherjee. Choosing multiple parameters for support vector machines. *Machine learning*, 46(1-3):131–159, 2002.

Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. Adanet: Adaptive structural learning of artificial neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 874–883. JMLR. org, 2017.

Sauptik Dhar, Vladimir Cherkassky, and Mohak Shah. Multiclass learning from contradictions. In *Advances in Neural Information Processing Systems*, pages 8400–8410, 2019.

Jonathan Eckstein and Dimitri P Bertsekas. On the douglas—rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318, 1992.

Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1165–1173. JMLR. org, 2017.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimilano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. *arXiv preprint arXiv:1806.04910*, 2018.

Pontus Giselsson and Stephen Boyd. Linear convergence and metric selection for douglas-rachford splitting and admm. *IEEE Transactions on Automatic Control*, 62(2):532–544, 2016.

Tom Goldstein, Brendan O'Donoghue, Simon Setzer, and Richard Baraniuk. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences*, 7(3):1588–1623, 2014.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

Jiayi Liu, Samarth Tripathi, Unmesh Kurup, and Mohak Shah. Auptimizer–an extensible, open-source framework for hyperparameter tuning. *arXiv preprint arXiv:1911.02522*, 2019.

J. Lorraine and D. Duvenaud. Stochastic hyperparameter optimization through hypernetworks. arXiv:1802.09419, 2018.

M. MacKay, P. Vicol, J. Lorraine, D. Duvenaud, and R. Grosse. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. In *International Conference on Learning Representations*, 2019.

Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International Conference on Machine Learning*, pages 2113–2122, 2015.

Akshay Mehra and Jihun Hamm. Penalty method for inversion-free deep bilevel optimization. *arXiv preprint arXiv:1911.03432*, 2019.

Brian G Osborne, Thomas Fearn, Andrew R Miller, and Stuart Douglas. Application of near infrared reflectance spectroscopy to the compositional analysis of biscuits and biscuit doughs. *Journal of the Science of Food and Agriculture*, 35(1):99–105, 1984.

Y. Park, S. Dhar, S. Boyd, and M. Shah. Variable metric proximal gradient method with diagonal barzilai-borwein stepsize. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3597–3601, 2020.

Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *Proceedings of the 33nd International Conference on Machine Learning, ICML*, 2016. URL `http://jmlr.org/proceedings/papers/v48/pedregosa16.html`.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, pages –, 2012. ISSN 0893-6080. doi: 10.1016/j.neunet.2012.02.016.

Paul Tseng. Applications of a splitting algorithm to decomposition in convex programming and variational inequalities. *SIAM Journal on Control and Optimization*, 29(1):119–138, 1991.

Appendix A. Proofs

A.1 Proof of Proposition 1

The proof follows from analyzing the KKT equations. Note that the lagrangian of the problem 2 is,

$$\mathcal{L}(\mathbf{w}, \lambda, \mathbf{u}) = L_T(\mathbf{w}, \lambda^*) + L_V(\lambda, G_{\phi^*}(\lambda)) + \mathbf{u}^T(\mathbf{w} - G_{\phi^*}(\lambda)) \tag{8}$$

<u>KKT system</u>,

$$\nabla_{\mathbf{w}} L_T(\mathbf{w}, \lambda^*) + \mathbf{u} = \qquad\qquad 0 \tag{9}$$
$$\nabla_\lambda L_V(\lambda, G_{\phi^*}(\lambda)) - \mathbf{u}^T \nabla_\lambda G_{\phi^*}(\lambda) = \qquad\qquad 0 \tag{10}$$
$$\mathbf{w} - G_{\phi^*}(\lambda) = \qquad\qquad 0 \tag{11}$$
$$\nabla_\phi L_T(G_{\phi^*}(\lambda), \lambda) = \qquad\qquad 0 \tag{12}$$

At the solution of the KKT System $(\mathbf{w}^*, \lambda^*, \mathbf{u}^*)$ for a bijective mapping $\mathbf{w}^* = G_{\phi^*}(\lambda^*) = \Lambda^* \phi^*$ we have eq. (12) $\Rightarrow \nabla_{\mathbf{w}} L_T(\mathbf{w}^*, \lambda^*) = 0$. This in turn indicates, $\mathbf{u}^* = 0 \Rightarrow \nabla_\lambda L_V(\lambda^*, G_{\phi^*}(\lambda^*)) = 0$. Hence at solution, $\nabla_{\mathbf{w}} L_T(\mathbf{w}^*, \lambda^*) = 0$ and $\nabla_\lambda L_V(\lambda^*, G_{\phi^*}(\lambda^*)) = 0$ ∎

A.2 Proof of Proposition 2

Assume the conditions $r^{k+1} = 0$ and $s^{k+1} = 0$, are satisfied at iteration $k + 1$. Then the steps $1 - 4$ gives,

$$\nabla_{\mathbf{w}} L_T(\mathbf{w}^{k+1}, \lambda^k) + \mathbf{u}^k + \rho(\mathbf{w}^{k+1} - \Lambda^k \phi^{k+1}) = 0 \tag{13}$$
$$\Rightarrow \nabla_{\mathbf{w}} L_T(\mathbf{w}^{k+1}, \lambda^k) + \mathbf{u}^{k+1} + \rho(\Lambda^{k+1} \phi^{k+1} - \Lambda^k \phi^{k+1}) = 0 \tag{14}$$
$$\nabla_\lambda L_V(\lambda^{k+1}, \Lambda^{k+1} \phi^{k+1}) - (\mathbf{u}^k)^T \phi_1^{k+1} - \rho(\phi_1^{k+1})^T(\mathbf{w}^{k+1} - \Lambda^{k+1} \phi^{k+1}) = 0 \tag{15}$$
$$\mathbf{u}^{k+1} = \mathbf{u}^k + \rho(\mathbf{w}^{k+1} - \Lambda^{k+1} \phi^{k+1}) \tag{16}$$

Under the conditions $r^{k+1} = 0, s^{k+1} = 0$ the updates in eq. (13) - (16) becomes,

$$\nabla_{\mathbf{w}} L_T(\mathbf{w}^{k+1}, \lambda^k) + \mathbf{u}^{k+1} = 0$$
$$\nabla_\lambda L_V(\lambda^{k+1}, \Lambda^{k+1} \phi^{k+1}) - (\mathbf{u}^{k+1})^T \phi_1^{k+1} = 0 \tag{17}$$
$$\mathbf{u}^{k+1} = \mathbf{u}^k$$

In addition at iteration $k + 1$ we also have from eq. (4) a unique solution for

$$(\phi_1^{k+1}, \phi_0^{k+1}) = \underset{\boldsymbol{\theta}_1, \boldsymbol{\theta}_0}{\operatorname{argmin}} L_T(\lambda^k \boldsymbol{\theta}_1 + \boldsymbol{\theta}_0, \lambda^k)$$

Also we have from our assumptions,

$$r^{k+1} = 0 \Rightarrow \mathbf{w}^{k+1} = \lambda^k \phi_1^{k+1} + \phi_0^{k+1} \Rightarrow \nabla_{\mathbf{w}} L_T(\mathbf{w}^{k+1}, \lambda^k) = 0 \tag{18}$$
$$s^{k+1} = 0 \Rightarrow \lambda^k \phi_1^{k+1} = \lambda^{k+1} \phi_1^{k+1} \Rightarrow \nabla_{\mathbf{w}} L_T(\mathbf{w}^{k+1}, \lambda^{k+1}) = 0 \tag{19}$$

Using e.q. (18) and (19) into (17) gives,

$$\nabla_{\mathbf{w}} L_T(\mathbf{w}^{k+1}, \lambda^{k+1}) = 0 \quad (\Rightarrow \mathbf{u}^{k+1} = 0)$$
$$\nabla_{\lambda} L_V(\lambda^{k+1}, \Lambda^{k+1}\boldsymbol{\phi}^{k+1}) = 0 \quad (\because \mathbf{u}^{k+1} = 0)$$
$$\Rightarrow \nabla_{\lambda} L_V(\lambda^{k+1}, \mathbf{w}^{k+1}) = 0 \quad (\text{for bijective } \Lambda\boldsymbol{\phi} \to \mathbf{w})$$

Next for the necessary part if $r^{k+1} \neq 0$, the primary constraint in (3) is not satisfied. Hence $r^{k+1} = 0$ is a necessary condition. To establish the necessary condition $s^{k+1} = 0$, consider $\nabla_{\lambda} L_V(\lambda^{k+1}, \mathbf{w}^{k+1}) = 0$ and $\nabla_{\mathbf{w}} L_T(\mathbf{w}^{k+1}, \lambda^{k+1}) = 0$.

Now, $r^{k+1} = 0 \Rightarrow \mathbf{w}^{k+1} = \boldsymbol{\phi}^{k+1}\Lambda^{k+1} \Rightarrow \nabla_{\boldsymbol{\phi}} L_T(\boldsymbol{\phi}^{k+1}\Lambda^{k+1}, \lambda^{k+1}) = 0$ (from above assumption). Finally, Step 1 of Algorithm 3 ensures, $\nabla_{\boldsymbol{\phi}} L_T(\boldsymbol{\phi}^{k+1}\Lambda^{k}, \lambda^{k}) = 0$. Hence $\boldsymbol{\phi}^{k+1}\Lambda^{k} = \boldsymbol{\phi}^{k+1}\Lambda^{k+1} \Rightarrow \mathbf{s}^{k+1} = 0$. $\blacksquare$

A.3 Proof for Claim 1

For the first part of the proof observe that the steps $2-4$ are exactly the ADMM updates for a given $\boldsymbol{\phi}^{k+1}$. This allows us to re-utilze the Proposition 4 in (Giselsson and Boyd, 2016) and claim that the operator equivalent to the steps $2-4$ is contractive. The exact form of this new operator and the equivalent convergence rates will be analyzed in a longer version of this work.

For the second part observe that the state of the system is determined by the variables $(\lambda^k, \mathbf{u}^k) \to (\lambda^{k+1}, \mathbf{u}^{k+1})$. At stationary points the states remain the same. This gives,

From (16), $\quad \mathbf{w}^{k+1} - \Lambda^{k+1}\boldsymbol{\phi}^{k+1} = 0 \Rightarrow r^{k+1} = 0$

Further, $\rho(\Lambda^{k+1}\boldsymbol{\phi}^{k+1} - \Lambda^{k}\boldsymbol{\phi}^{k+1}) = 0 \qquad (\because \Lambda^{k} = \Lambda^{k+1} \text{and } \phi \text{ is bounded })$

$\blacksquare$

Appendix B. Stochastic Hyperparameter Optimization using Hypernetworks (SHO)

There are several versions (global vs. local) of the SHO algorithm introduced in (Lorraine and Duvenaud, 2018). The representative local version of the SHO algorithm is provided below in Algorithm 2.

Algorithm 2: SHO Algorithm (Local)

Input:

$\quad\quad\alpha\quad$ =learning rate of training loss gradient update w.r.t model parameters.

$\quad\quad\beta\quad$ =learning rate of validation loss gradient update w.r.t hyperparameters.

Output: $\lambda, \mathbf{G}_\phi(\lambda)$

Data: $\mathcal{T}$ = Training Data , $\mathcal{V}$ = Validation Data

Initialize: ϕ, λ and, **define**: $\mathbf{w} = G_\phi(\lambda) = \Lambda\,\phi$;

while *not_converged* **do**

$\quad\quad \hat{\lambda} \sim P(\hat{\lambda}|\lambda^{t-1})$ // typically modeled as Normal Distribution

$\quad\quad \phi^t \leftarrow \phi^{t-1} - \alpha \nabla_{G_\phi} L_T(G_\phi(\hat{\lambda}), \hat{\lambda}, \mathbf{x} \in \mathcal{T}) \cdot \nabla_\phi G_\phi(\hat{\lambda})$;

$\quad\quad \lambda^t \leftarrow \lambda^{t-1} - \beta \underbrace{\nabla_\lambda G_\phi(\lambda)}_{=\phi^T} \cdot \overset{=\hat{\Lambda}}{\nabla_{G_\phi} L_V(G_\phi(\lambda), \mathbf{x} \in \mathcal{V})}$

end

Appendix C. Moreau Yosida regularized (MY)-HPO Algorithm

Algorithm 3: MY-HPO algorithm

Input: MaxIters = Total iterations, $\quad \varepsilon_{TOL}$ = convergence error tolerance.

Output: $\lambda, \mathbf{w}$

Data: $\mathcal{T}$ = Training Data , $\mathcal{V}$ = Validation Data

Initialize: $\lambda \leftarrow -1$ and $\mathbf{u} \leftarrow \mathbf{0}$

while $(k + 1 \leq$ MaxIters $)$ and $($ *convergence_error* $\geq \varepsilon_{TOL}$ $)$ **do**

$\quad\quad$**Step 1.** Update ϕ

$$\mathbf{v}^{k+1} = \underset{\mathbf{v}}{\arg\min}\ L_T(\mathbf{v}, \lambda^k)$$

$$\phi_0^{k+1} = \overline{\mathbf{v}^{k+1}} \quad ; \quad \phi_1^{k+1} = \frac{\mathbf{v}^{k+1} - \overline{\mathbf{v}^{k+1}}}{\lambda^k} \quad \text{and} \quad \phi^{k+1} = [\phi_1^{k+1}|\phi_0^{k+1}]$$

$\quad\quad$**Step 2.** Update $\mathbf{w}$

$$\mathbf{w}^{k+1} = \underset{\mathbf{w}}{\arg\min}\ L_T(\mathbf{w}, \lambda^k) + (\mathbf{u}^k)^T(\mathbf{w} - \Lambda^k \phi^{k+1}) + \frac{\rho}{2}||\mathbf{w} - \Lambda^k \phi^{k+1}||_2^2$$

$\quad\quad$**Step 3.** Update λ

$$\lambda^{k+1} = \underset{\lambda}{\arg\min}\ L_V(\lambda, G_{\phi^{k+1}}(\lambda)) + (\mathbf{u}^k)^T(\mathbf{w}^{k+1} - \Lambda \phi^{k+1}) + \frac{\rho}{2}||\mathbf{w}^{k+1} - \Lambda \phi^{k+1}||_2^2$$

$\quad\quad$**Step 4.** Update consensus $\mathbf{u}^{k+1} = \mathbf{u}^k + \rho(\mathbf{w}^{k+1} - \Lambda^{k+1} \phi^{k+1})$

end

Connection with ADMM : The MY-HPO updates are closely connected to the Alternating Direction Method of Multipliers (ADMM) algorithm (Boyd et al., 2011). In fact, for a given ϕ^k the steps $2 - 4$ are exactly the ADMM updates. However, the algorithm is fundamentally different from ADMM. For example, unlike ADMM, interchanging the steps 2 and 3 completely changes the stationary points. Moreover, the updates do not transform

Table 3: Performance of MY-HPO Algorithm 3 for the regression problems in Table 2

Data	Train Loss	Val. Loss	Test Loss	No. of Iterations (k)	MaxIters
Cookie	0.31 ± 0.20	0.22 ± 0.21	0.26 ± 0.25	80 ± 18.68	100
MNIST	8.85 ± 0.35	9.85 ± 0.47	9.32 ± 0.35	364 ± 124.3	500

to the Douglas Rachford splitting operator as traditionally seen for ADMM (Giselsson and Boyd, 2016). Still, the similarities of MY-HPO with ADMM enables us to modify previous convergence analyses in (Giselsson and Boyd, 2016) to obtain Proposition 2 and Claim 1.

Relaxation to simplified MY-HPO algorithm 1 : Algorithm 1 simplifies the above algorithm 3 by taking one gradient update rather than solving the minimization problems above. This reduces the per-step iteration cost of the algorithm 1 to be the same as the SHO algorithm in 2. Now, both the Algorithms 1 and 2 incurs 2-gradient steps per outer iteration. This simplification however deteriorates the convergence rate of the algorithm. We provide the convergence comparison between the Algorithms 1 vs. 3 in Table 3.

Appendix D. Algorithm Parameters and Additional Results

For all the black-box approaches we match the number of outer-iterations of the bi-level formulations i.e. n_T to train the model for any given parameter. Further, we train the models using gradient updates with step size (α_{train}). This parameter is set specific to each problems reported below, and helps us achieve similar training gradient as the bi-level formulations. In addition, we set the following parameters using the Auptimizer tool-box for each of the following algorithms.

<u>Random Search</u>: We select the *random_seed* same as that used to generate the data. Additionally, we search in the range $[-10, \cdots, 5]$. We report the performance of the algorithm for varying number of search parameters n_S.

<u>Grid Search</u>: We select the range of search as, $[-10, \cdots, 5]$. We report the performance of the algorithm for varying number of search parameters n_S.

<u>HyperOpt</u>: We select the *random_seed* same as that used to generate the data. We select the range of search as, $[-10, \cdots, 5]$ and the engine = 'tpe'. We report the algorithm's performance for varying values of n_S.

<u>Spearmint</u>: We select the range of search as, $[-10, \cdots, 5]$, engine = 'GPEIOptChooser' and *grid_size* = 20000. We report the algorithm's performance for varying values of n_S.

The parameters specific to the problems and the respective data-sets are provided below.

D.1 Regression using Cookie Data

D.1.1 Experiment Parameters

Stochastic Hypernetwork Optimization (SHO): We keep the following parameter fixed (to default values), $\beta = 0.01$, noise variance $\hat{\lambda} \sim \mathcal{N}(\lambda, \sigma), \sigma = 10^{-4}$. Changing these values for our experiments did not result in significant improvements. We report the performance

of SHO for varying values of α. The code is publicly available at: `https://github.com/lorraine2/hypernet-hypertraining`

Moreau Yosida Regularized HPO (MY-HPO): We keep the following parameter fixed, $\rho = 1.0$ and report the results for varying α, β, δ. It is well-known in ADMM literature that the selection of $\rho-$ parameter greatly impacts the convergence behavior. Such analyses will be explored in a longer version of the paper.

Further we fix n_T (i.e. MaxIters in Algorithm 3) = 2500, and step size for training the model used for black-box approaches as $\alpha_{train} = 0.001$.

D.1.2 Additional Results

The complete set of results with all the parameter settings are provided in Table 4. As seen from the results increasing the budget (number of gradient computations n_g) to almost $5\times$ that of the bi-level counterparts allows the black-box approaches achieve similar performance.
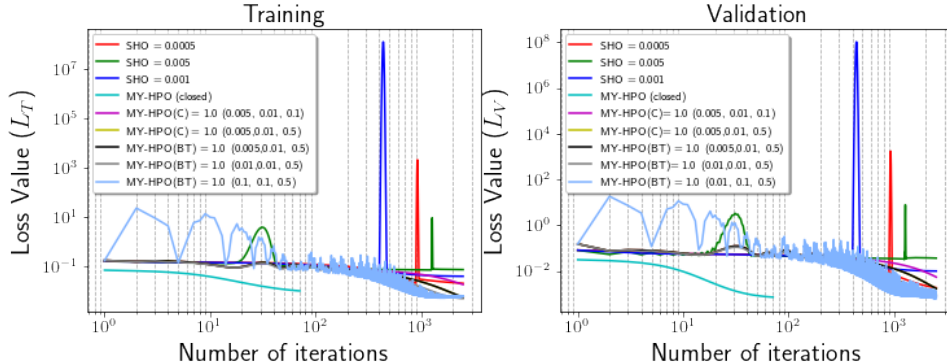

Figure 2: Convergence behavior of SHO vs. MY-HPO for different step-sizes for Cookie data using Least Square loss. For MY-HPO we show the $\rho(\alpha, \beta, \delta)$ values.

The figure 2 illustrate the convergence curve of the algorithms. Fig. 2 illustrates the unstable behavior of SHO as the step size increases. Note that, for larger step sizes the SHO algorithm fails to converge. As the step size gets smaller the algorithm converges; but demonstrate a very slow convergence rate. This is also seen for the other experiments used in this paper. On the contrary MY-HPO can accomodate higher step size and results to better convergence. Additionally, 2 also provides a comparison between the MY-HPO algorithm in 3 vs. the simplified version of algorithm 1. As seen in the figure (also confirmed in Table 3), the simplified algorithm provides better per-step iteration cost but incurs much higher overall iterations to convergence.

13

Table 4: Comparison between different HPO algorithms for regression problem using Cookie data.

METHOD	TRAIN LOSS $(\times 10^-2)$	VALIDATION LOSS $(\times 10^-2)$	TEST LOSS $(\times 10^-2)$	n_g (n_T, n_S)
SHO ($\alpha = 5 \times 10^{-2}$)	125.18 ± 17.08	157.99 ± 33.43	143.32 ± 40.32	5000 $(2500, 2)$
SHO ($\alpha = 1 \times 10^{-2}$)	86.7 ± 56.08	80.76 ± 62.36	85.97 ± 88.78	5000 $(2500, 2)$
SHO ($\alpha = 5 \times 10^{-3}$)	71.6 ± 66.96	63.18 ± 72.85	75.31 ± 98.78	5000 $(2500, 2)$
MYHPO (C) ($\alpha = 0.005, \beta = 0.01, \delta = 0.1$)	22.9 ± 15.51	28.27 ± 19.94	30.6 ± 29.05	5000 $(2500, 2)$
MYHPO (C) ($\alpha = 0.005, \beta = 0.01, \delta = 0.5$)	7.8 ± 2.68	17.64 ± 13.41	17.42 ± 15.82	5000 $(2500, 2)$
MYHPO (BT) ($\alpha = 0.005, \beta = 0.01, \delta = 0.5$)	7.8 ± 2.68	17.64 ± 13.41	17.42 ± 15.82	5000 $(2500, 2)$
MYHPO (BT) ($\alpha = 0.01, \beta = 0.01, \delta = 0.5$)	6.85 ± 5.68	9.74 ± 6.6	11.31 ± 13.54	5000 $(2500, 2)$
MYHPO (BT) ($\alpha = 0.1, \beta = 0.1, \delta = 0.5$)	5.41 ± 2.53	6.28 ± 4.82	6.91 ± 7.25	5000 $(2500, 2)$
RANDOM ($n_S = 2$)	89.02 ± 10.36	93.44 ± 26.4	85.69 ± 28.97	5000 $(2500, 2)$
($n_S = 10$)	22.7 ± 4.18	22.56 ± 15.03	21.96 ± 17.05	25000 $(2500, 10)$
GRID ($n_S = 2$)	6.7 ± 2.21	16.71 ± 12.91	16.15 ± 14.76	5000 $(2500, 2)$
($n_S = 10$)	6.7 ± 2.21	16.71 ± 12.91	16.15 ± 14.76	25000 $(2500, 10)$
HYPEROPT ($n_S = 2$)	58.11 ± 7.97	49.77 ± 19.57	47.29 ± 23.54	5000 $(2500, 2)$
($n_S = 10$)	6.89 ± 2.19	16.75 ± 12.94	16.20 ± 14.79	25000 $(2500, 2)$
SPEARMINT ($n_S = 2$)	6.70 ± 2.21	16.71 ± 12.91	16.15 ± 14.76	5000 $(2500, 2)$
($n_S = 10$)	6.70 ± 2.21	16.71 ± 12.91	16.15 ± 14.76	25000 $(2500, 10)$

D.2 Regression using MNIST Data

D.2.1 EXPERIMENT PARAMETERS

Stochastic Hypernetwork Optimization (SHO): We fix the parameters same as in D.1.1. We vary the α parameter as shown in Table 5.

Moreau Yosida Regularized HPO (MY-HPO): We fix the parameters same as in D.1.1. We vary the α, β, δ parameters as shown in Table 5.

Further for this data we fix n_T (i.e. MAXITERS in Algorithm 3) $= 3000$, and step size for training the model used for black-box approaches as $\alpha_{train} = 0.001$.

Table 5: Comparison between different HPO algorithms for regression problem using MNIST data.

METHOD	TRAIN LOSS $(\times 10^{-}2)$	VALIDATION LOSS $(\times 10^{-}2)$	TEST LOSS $(\times 10^{-}2)$	n_g (n_T, n_S)
SHO ($\alpha = 1 \times 10^{-2}$)	32.67 ± 1.83	32.27 ± 6.13	32.33 ± 7.46	6000 (3000, 2)
SHO ($\alpha = 5 \times 10^{-3}$)	27.28 ± 3.69	25.26 ± 1.22	24.22 ± 0.98	6000 (3000, 2)
SHO ($\alpha = 1 \times 10^{-3}$)	26.7 ± 2.79	24.63 ± 1.37	23.45 ± 1.11	6000 (3000, 2)
MYHPO (C) ($\alpha = 0.001, \beta = 0.001, \delta = 0.001$)	23.13 ± 0.66	23.67 ± 0.97	22.31 ± 0.48	6000 (3000, 2)
MYHPO (C) ($\alpha = 0.001, \beta = 0.001, \delta = 0.005$)	22.63 ± 0.63	23.69 ± 1.02	22.32 ± 0.55	6000 (3000, 2)
MYHPO (BT) ($\alpha = 0.001, \beta = 0.001, \delta = 0.005$)	22.63 ± 0.63	23.69 ± 1.02	22.32 ± 0.55	6000 (3000, 2)
MYHPO (BT) ($\alpha = 0.01, \beta = 0.01, \delta = 0.05$)	22.43 ± 0.61	23.69 ± 1.11	22.53 ± 0.72	6000 (3000, 2)
MYHPO (BT) ($\alpha = 0.1, \beta = 0.1, \delta = 0.5$)	21.75 ± 0.77	24.16 ± 1.18	22.85 ± 0.89	6000 (3000, 2)
RANDOM ($n_S = 2$)	22.28 ± 6.04	25.59 ± 2.86	24.13 ± 2.85	6000 (3000, 2)
GRID ($n_S = 2$)	17.44 ± 0.52	26.3 ± 2.48	24.79 ± 2.17	6000 (3000, 2)
HYPEROPT ($n_S = 2$)	22.84 ± 5.75	25.02 ± 2.48	23.85 ± 2.15	6000 (3000, 2)
SPEARMINT ($n_S = 2$)	17.44 ± 0.52	26.30 ± 2.48	24.79 ± 2.17	6000 (3000, 2)
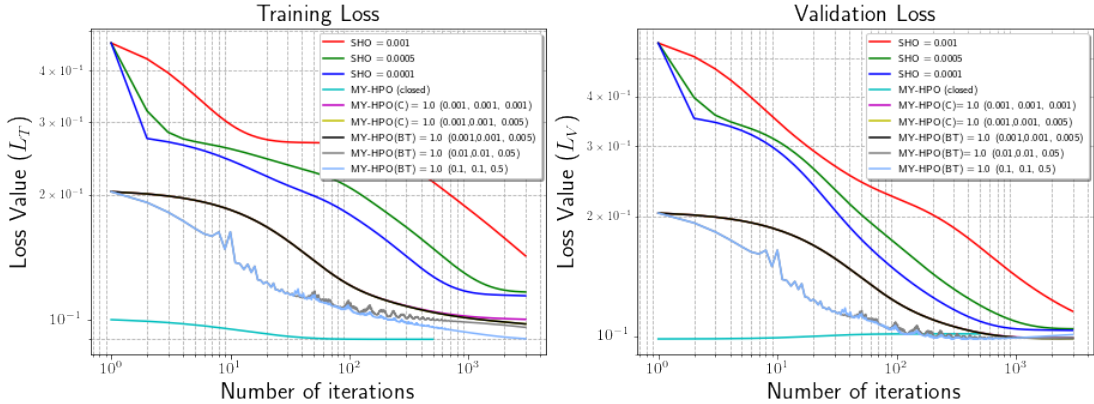
Figure 3: Convergence behavior of SHO vs. MY-HPO for different step-sizes for MNIST data using Least Square loss. For MY-HPO we show the $\rho(\alpha, \beta, \delta)$ values.

15

D.2.2 ADDITIONAL RESULTS

Note that, here further increasing the computation budget (i.e. $n_S = 10$) does not provide additional improvement for the black-box approaches. The results provide similar conclusions,

1. MY-HPO accomodates for higher step sizes and results to better convergence.
2. Back-tracking further ensures descent direction (in each iteration) and leads to better convergence.
3. Increasing the budget (number of gradient computations n_g) to almost $5\times$ that of the bi-level counterparts allows the black-box approaches achieve similar performance.

D.3 Classification using MNIST Data

D.3.1 EXPERIMENT PARAMETERS

Stochastic Hypernetwork Optimization (SHO): We fix the parameters same as in D.1.1. We vary the α parameter as shown in Table 6.

Moreau Yosida Regularized HPO (MY-HPO): We fix the parameters same as in D.1.1. We vary the α, β, δ parameters as shown in Table 6.

Further for this data we fix n_T (i.e. MAXITERS in Algorithm 3) $= 500$, and step size for training the model used for black-box approaches as $\alpha_{train} = 0.5$.
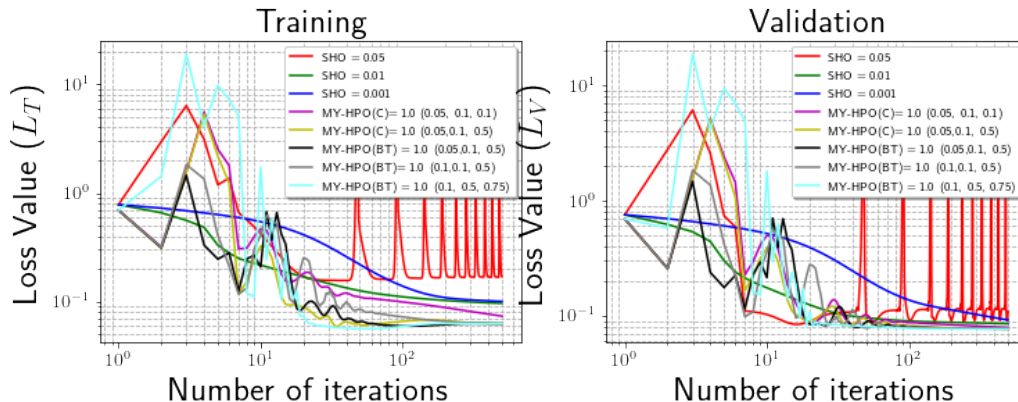
D.3.2 ADDITIONAL RESULTS

Figure 4: Convergence behavior of SHO vs. MY-HPO for different step-sizes for MNIST data using Logistic loss. For MY-HPO we show the $\rho(\alpha, \beta, \delta)$ values.

Table 6: Comparison between different HPO algorithms for classification problem using MNIST data.

METHOD	TRAIN LOSS $(\times 10^{-2})$	VALIDATION LOSS $(\times 10^{-2})$	TEST LOSS $(\times 10^{-2})$	n_g (n_T, n_S)
SHO ($\alpha = 5 \times 10^{-2}$)	47.46 ± 74.13	31.19 ± 71.85	31.56 ± 72.45	1000 $(500, 2)$
SHO ($\alpha = 1 \times 10^{-2}$)	5.58 ± 2.14	4.98 ± 1.96	5.32 ± 1.6	1000 $(500, 2)$
SHO ($\alpha = 1 \times 10^{-3}$)	6.58 ± 1.93	5.59 ± 1.75	5.68 ± 1.80	1000 $(500, 2)$
MYHPO (C) ($\alpha = 0.05, \beta = 0.1, \delta = 0.1$)	5.17 ± 0.97	4.91 ± 1.65	5.23 ± 1.49	1000 $(500, 2)$
MYHPO (C) ($\alpha = 0.05, \beta = 0.1, \delta = 0.5$)	2.99 ± 1.57	4.42 ± 1.89	5.06 ± 1.83	1000 $(500, 2)$
MYHPO (BT) ($\alpha = 0.05, \beta = 0.1, \delta = 0.5$)	2.94 ± 1.60	4.40 ± 1.89	5.06 ± 1.84	1000 $(500, 2)$
MYHPO (BT) ($\alpha = 0.1, \beta = 0.1, \delta = 0.5$)	3.17 ± 1.47	4.40 ± 1.88	5.02 ± 1.72	1000 $(500, 2)$
MYHPO (BT) ($\alpha = 0.1, \beta = 0.5, \delta = 0.75$)	2.87 ± 1.68	4.35 ± 1.94	5.06 ± 1.85	1000 $(500, 2)$
RANDOM ($n_S = 2$)	35.733 ± 71.443	19.07 ± 37.81	19.49 ± 38.62	1000 $(500, 2)$
($n_S = 25$)	2.17 ± 0.93	4.46 ± 2.03	5.05 ± 1.89	12500 $(500, 25)$
GRID ($n_S = 2$)	0.3 ± 0.1	5.31 ± 2.71	6.22 ± 2.82	1000 $(500, 2)$
($n_S = 25$)	2.53 ± 1.23	4.37 ± 1.88	5.05 ± 1.84	12500 $(500, 25)$
HYPEROPT ($n_S = 2$)	5.18 ± 0.35	4.91 ± 1.46	5.19 ± 1.44	1000 $(500, 2)$
($n_S = 25$)	2.44 ± 1.43	4.40 ± 1.88	4.18 ± 1.97	12500 $(500, 25)$
SPEARMINT ($n_S = 2$)	0.30 ± 0.10	5.31 ± 2.71	6.22 ± 2.82	1000 $(500, 2)$
($n_S = 25$)	1.89 ± 1.55	4.63 ± 2.19	5.36 ± 2.27	12500 $(500, 25)$

D.4 Classification using GTSRB Data

D.4.1 EXPERIMENT PARAMETERS

Stochastic Hypernetwork Optimization (SHO): We fix the parameters same as in D.1.1. We vary the α parameter as shown in Table 6.

Moreau Yosida Regularized HPO (MY-HPO): We fix the parameters same as in D.1.1. We vary the α, β, δ parameters as shown in Table 6.

Further for this data we fix n_T (i.e. MAXITERS in Algorithm 3) = 500, and step size for training the model used for black-box approaches as $\alpha_{train} = 0.5$.

D.4.2 ADDITIONAL RESULTS

Table 7: Comparison between different HPO algorithms for classification problem using GTSRB data.

METHOD	TRAIN LOSS $(\times 10^{-2})$	VALIDATION LOSS $(\times 10^{-2})$	TEST LOSS $(\times 10^{-2})$	n_g (n_T, n_S)
SHO ($\alpha = 5 \times 10^{-2}$)	27.48 ± 20.54	25.38 ± 19.09	25.61 ± 20.27	1000 $(500, 2)$
SHO ($\alpha = 1 \times 10^{-2}$)	6.83 ± 1.04	14.35 ± 2.34	14.04 ± 1.82	1000 $(500, 2)$
SHO ($\alpha = 5 \times 10^{-3}$)	6.89 ± 1.13	14.44 ± 2.41	14.14 ± 1.85	1000 $(500, 2)$
MYHPO (C) ($\alpha = 0.05, \beta = 0.1, \delta = 0.1$)	7.98 ± 0.74	14.81 ± 2.2	14.53 ± 1.61	1000 $(500, 2)$
MYHPO (C) ($\alpha = 0.05, \beta = 0.1, \delta = 0.75$)	3.26 ± 2.50	14.14 ± 2.38	13.80 ± 2.25	1000 $(500, 2)$
MYHPO (BT) ($\alpha = 0.05, \beta = 0.1, \delta = 0.75$)	3.26 ± 2.50	14.14 ± 2.38	13.80 ± 2.25	1000 $(500, 2)$
MYHPO (BT) ($\alpha = 0.1, \beta = 0.5, \delta = 0.75$)	4.37 ± 2.11	14.09 ± 2.36	13.77 ± 2.11	1000 $(500, 2)$
MYHPO (BT) ($\alpha = 0.5, \beta = 0.5, \delta = 0.75$)	4.31 ± 1.22	13.98 ± 2.33	13.59 ± 1.98	1000 $(500, 2)$
RANDOM ($n_S = 2$)	47.48 ± 117.38	39.85 ± 75.38	38.60 ± 70.63	1000 $(500, 2)$
($n_S = 10$)	4.01 ± 2.07	14.19 ± 2.24	13.63 ± 1.91	5000 $(500, 10)$
GRID ($n_S = 2$)	0.09 ± 0.01	23.55 ± 6.60	21.81 ± 5.23	1000 $(500, 2)$
($n_S = 10$)	3.53 ± 0.49	13.87 ± 2.34	13.40 ± 2.03	5000 $(500, 10)$
HYPEROPT ($n_S = 2$)	11.91 ± 30.50	18.9 ± 11.7	17.62 ± 8.52	1000 $(500, 2)$
($n_S = 10$)	3.3 ± 1.46	13.94 ± 2.40	13.58 ± 2.17	5000 $(500, 10)$
SPEARMINT ($n_S = 2$)	0.09 ± 0.01	23.55 ± 6.60	21.81 ± 5.23	1000 $(500, 2)$
($n_S = 10$)	2.81 ± 5.63	19.73 ± 6.28	19.20 ± 5.59	5000 $(500, 10)$