OPT2024: 16th Annual Workshop on Optimization for Machine Learning

Aligned Multi Objective Optimization

Yonathan Efroni *Meta AI*
Daniel Jiang *Meta AI*
Ben Kretzu *Technion*
Jalaj Bhandari *Meta AI*
Zheqing (Bill) Zhu *Meta AI*
Karen Ullrich *Meta AI*

Abstract

To date, the multi-objective optimization literature has mainly focused on conflicting objectives, studying the Pareto front or requiring users to balance tradeoffs. Yet, in machine learning practice, there are many scenarios where such conflict does not take place. Recent findings from multi-task learning, reinforcement learning, and LLMs show that diverse related tasks can enhance performance across objectives simultaneously. Despitechcuh this evidence, such phenomenon has not been examined from an optimization perspective. This leads to a lack of generic gradient-based methods that can scale to scenarios with a large number of related objectives. To address this gap, we introduce the Aligned Multi-Objective Optimization framework, propose the `AMOOO` algorithm, and provide theoretical guarantees of its superior performance compared to naive approaches.

1. Introduction

In many real-world optimization problems, we have access to multi-dimensional feedback rather than a single scalar objective. The multi-objective optimization (MOO) literature has largely focused on the setting where these objectives *conflict* with each other, which necessitates the *Pareto dominance* notion of optimality. A closely related area of study is *multi-task learning* [31], where multiple tasks are learned jointly, typically with both shared and task-specific parameters. The hope is that the model can perform better on individual task by sharing common information across tasks. Indeed, the phenomenon of improved performance across all tasks has been observed in several settings [17, 20], suggesting that perhaps there may not be significant trade-offs between objectives.

In this paper, we explicitly consider a setting where objectives are *aligned*, i.e., objectives that share a common solution. For example, in reinforcement learning, practitioners can sometimes speed up learning by exploit several alternative reward specifications that all lead to the same optimal policy [8]. In statistics and machine learning, labeled data is sometimes sparse, leading practitioners to rely on closely-related proxy tasks to improve prediction accuracy [2].

To our knowledge, there is no work that studies this setting from a purely optimization perspective. We ask the question: *how can an optimization algorithm benefit from multi-objective feedback when the objectives are aligned?* We introduce the *aligned multi-objective optimization* (AMOO) framework to study this question. Subsequently, we design a new algorithm with provable guarantees for the AMOO setting and show empirical evidence of improved convergence properties.

2. Aligned Multi Objective Optimization

Consider an unconstrained multi-objective optimization where $F : \mathbb{R}^n \to \mathbb{R}^m$ is a vector valued function, $F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}))$, and all functions $\{f_i\}_{i\in[m]}$ are convex where $[m] := \{1, \ldots, m\}$. Without additional assumptions the components of $F(\mathbf{x})$ cannot be minimized simultaneously. To define a meaningful approach to optimize $F(\mathbf{x})$ one can study the Pareto front, or to properly define how to trade-off the objectives. We denote by Δ_m the m-dimensional simplex, and by $\Delta_{m,\alpha} := \{\mathbf{w} \in \mathbb{R}^m : \mathbf{w} \in \Delta_m, \forall i \in [m]\ w_i \geq \alpha\}$. In the AMOO setting we make the assumption the functions are aligned in a specific sense: we assume that the functions $\{f_i\}_{i\in[m]}$ share an optimal solution. Namely, there exists a point $\mathbf{x}^*$ that minimizes all functions in $F(\cdot)$ simultaneously,

$$\mathbf{x}^* \in \arg\min_{x\in\mathbb{R}^n} f_i(\mathbf{x}) \quad \forall i \in [m]. \tag{1}$$

With this additional assumption one may hope to get quantitative benefits from the multi objective feedback. How can Gradient Descent (GD) be improved when the functions are aligned?

A common algorithmic approach in the multi-objective setting is using a weight vector $\mathbf{w} \in \mathbb{R}^m$ that maps the vector $F(\mathbf{x})$ into a single objective $f_{\mathbf{w}}(\mathbf{x}) := \mathbf{w}^T F(\mathbf{x})$, amenable to GD optimization [22, 25, 31, 37]. Existing algorithms suggest alternatives for choosing $\mathbf{w}$. We follow this paradigm and design an algorithm that chooses the weights adaptively for the AMOO setting.

Towards developing intuition for our algorithmic approach we consider few examples of the AMOO setting. These showcase the need to choose weights in an adaptive way to the problem.

The Specification Example. Consider the case $F(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}))$, $\mathbf{x} \in \mathbb{R}^2$ where

$$f_1(\mathbf{x}) = (1-\Delta)x_1^2 + \Delta x_2^2, \quad \text{and} \quad f_2(\mathbf{x}) = \Delta x_1^2 + (1-\Delta)x_2^2,$$

for some small $\Delta \in [0, 0.5]$. It is clear that $F(\mathbf{x})$ can be simultaneously minimized in $\mathbf{x}_\star = (0,\ 0)$, hence, this is an AMOO setting. This example, as we demonstrate, illustrates an instance in which each individual function *does not specify the solution well*, but with proper weighting the optimal solution is well specified.

First, observe both f_1 and f_2 are Δ-strongly convex and $O(1)$-smooth functions. Hence, GD with properly tuned learning rate, applied to either f_1 or f_2 will converge with linear rate of $\Omega(\Delta)$. It is simple to observe this rate can be dramatically improved by proper weighting of the functions. Indeed, let $f_{\mathbf{w}_U}$ be a function with equal weighting of both f_1 and f_2, namely, choosing $\mathbf{w}_U = (0.5, 0.5)$, we get $f_{\mathbf{w}_U}(\mathbf{x}) = 0.5x_1^2 + 0.5x_2^2$ which is $\Omega(1)$-strongly convex and $O(1)$-Lipchitz smooth. Hence, GD applied to $f_{\mathbf{w}_U}$ converges with linear rate of $\Omega(1)$—much faster than $O(\Delta)$ when Δ is taken to be arbitrarily small.

The Selection Example. Consider the case $F(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))$, $\mathbf{x} \in \mathbb{R}^m$, where

$$\forall i \in [m-1] : f_i(\mathbf{x}) = (1-\Delta)x_1^2 + \Delta \sum_{j=2}^{d} x_j, \quad \text{and} \quad f_m(\mathbf{x}) = \sum_{j=1}^{d} x_j^2,$$

and $\Delta \in [0, 0.5]$. The common minimizer of all functions is $\mathbf{x}_\star = \mathbf{0} \in \mathbb{R}^d$, and, hence, the objectives are aligned. Unlike the specification example, in the selection example, there is a single objective function among the m objectives we should select to improve the convergence rate of GD. Further, in the selection example, choosing the uniform weight degrades the convergence rate.

Indeed, setting the weight vector to be uniform $\mathbf{w}_U = (1/m, \ldots, 1/m) \in \mathbb{R}^m$ leads to the function $f_{\mathbf{w}_U}(\mathbf{x}) = (2-\Delta)/m \cdot x_1^2 + \sum_{j=2}^{d}(\Delta+1)/m \cdot x_j^2$, which is $O(1/m)$-strongly convex. Hence, GD applied to $f_{\mathbf{w}_U}$ converges in a linear rate of $O(1/m)$. On the other hand, GD applied to f_m converges with linear rate of $\Omega(1)$. Namely, setting the weight vector to be $(0, \ldots, 0, 1) \in \mathbb{R}^m$ improves upon taking the average when the number of objectives is large.

Algorithm 1: AMOOO-GD

while $t = 1, 2, \ldots$ **do**
  $\mathbf{w}_t \leftarrow \text{AMOOO}\left(\{f_i(\mathbf{x}_t)\}_{i=1}^m\right)$
  $\mathbf{g}_t \leftarrow \nabla f_{\mathbf{w}_t}(\mathbf{x}_t)$
  $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \mathbf{g}_t$
end

Algorithm 2: AMOOO

inputs: $\{f_i(\mathbf{x}_t)\}_{i=1}^m$
initialize: $w_{\min} = \mu_\star / (8m\beta)$
Get Hessian matrices $\{\nabla^2 f_i(\mathbf{x}_t)\}_{i=1}^m$
$\mathbf{w}_t \in \arg\max_{\mathbf{w} \in \Delta_{m,w_{\min}}} \lambda_{\min}\left(\sum_i w_i \nabla^2 f_i(\mathbf{x}_t)\right)$
Return $\mathbf{w}_t$

3. Optimal Adaptive Strong Convexity & The AMOOO Algorithm

The aforementioned instances highlighted that in the AMOO setting the weights should be chosen in an adaptive way to the problem instance, and, specifically, based on the curvature. We formalize this intuition and design the AMOO-Optimizer (AMOOO). Towards developing it, we define the optimal adaptive strong convexity parameter, $\mu_\star$. Later we show that when the weighted loss is determined by AMOOO GD converges in a rate that depends on $\mu_\star$.

We start by defining the optimal adaptive strong convexity over the class of weights:

Definition 1 (Optimal Adaptive Strong Convexity $\mu_\star$) *The optimal adaptive strong convexity parameter, $\mu_\star \in \mathbb{R}_+$, is the largest value such that $\forall \mathbf{x} \in \mathcal{X}$ exists a weight vector $\mathbf{w} \in \Delta_m$ such that*

$$\lambda_{\min}\left(\sum_{i=1}^{m} w_i \nabla^2 f_i(\mathbf{x})\right) \geq \mu_\star. \tag{2}$$

For each $\mathbf{x} \in \mathcal{X}$, there may be a different weight vector in class $w_\star(\mathbf{x}) \in \Delta_m$ that solves $w_\star(\mathbf{x}) \in \arg\max \lambda_{\min}\left(\nabla^2 f_{\mathbf{w}}(\mathbf{x})\right)$ and maximizes the curvature. The optimal adaptive strong convexity parameter $\mu_\star$ is the largest lower bound on this quantity on the entire space $\mathcal{X}$. The specification and selection examples (Section 2) demonstrate $\mu_\star$ can be much larger than both the strong convexity parameter of the average function or of each individual function; for both $\mu_\star = O(1)$ whereas the alternatives may have arbitrarily small strongly convex parameter value.

Definition 1 not only quantifies an optimal notion of curvature, but also directly results with the AMOOO algorithm. AMOOO sets the weights according to equation 2, namely, at the k^{th} iteration, it finds the weight for which $f_{\mathbf{w}}(\mathbf{x}_k)$ has the largest local curvature. Then, a gradient step is applied in the direction of $\nabla f_{\mathbf{w}_k}(\mathbf{x}_k)$ (see Algorithm 1). Indeed, AMOOO seems as a natural candidate for AMOO. One may additionally hope that standard GD analysis techniques for strongly-convex and smooth functions can be applied. It is well known that if a function $f(\mathbf{x})$ is β smooth and $\forall \mathbf{x} \in \mathcal{X}$, $\lambda_{\min}\left(\nabla^2 f(\mathbf{x})\right) \geq \mu$ then GD converges with μ/β linear rate.

A careful examination of this argument shows it fails. Even though $\lambda_{\min}\left(\nabla^2 f_{\mathbf{w}_k}(\mathbf{x}_k)\right) \geq \mu_\star$ at each iteration it does not imply that $f_{\mathbf{w}_k}$ is $\mu_\star$ strongly convex for a fixed $\mathbf{w}_k$. Namely, it does not necessarily hold that for all $\mathbf{x} \in \mathcal{X}$, $\lambda_{\min}\left(\nabla^2 f_{\mathbf{w}_k}(\mathbf{x})\right) \geq \mu_\star$, but only pointwise at $\mathbf{x}_k$. This property emerges naturally in AMOO, yet such nuance is inherently impossible in single-objective optimization and, to the best of our knowledge, was not explored in online optimization as well.

Next, we provide a positive result. When restricting the class of functions to the set of self-concordant and smooth functions (see Appendix B) we provide a convergence guarantee for AMOOO-GD that depends on $\mu_\star$. The result shows that close to the optimal solution AMOOO-GD converges linearly with rate of $O(\mu_\star/\beta)$.

Theorem 2 ($\mu_\star$ **Convergence of AMOOO-GD**) *Suppose* $\{f_i\}_{i\in[m]}$ *are* β *smooth,* M_f *self-concordant, share an optimal solution* $\mathbf{x}_\star$ *and that* $\mu_\star > 0$. *Let* $k_0 := \lceil \left(\|\mathbf{x}_0 - \mathbf{x}_\star\| 3M_f\sqrt{m}\beta^2 - \beta\right)/\mu_\star^{3/2} \rceil$, *where* $\|\cdot\|$ *is the 2-norm. If* $\eta_t = 1/\beta$ *then* AMOOO-GD *converges with rate:*

$$\|\mathbf{x}_t - \mathbf{x}_\star\| \leq \begin{cases} (1-\mu_\star/8\beta)^{(k-k_0)/2}\sqrt{\mu_\star}/3M_f\sqrt{m}\beta & k \geq k_0 \\ \|\mathbf{x}_0 - \mathbf{x}_\star\| - k\mu_\star^{3/2}/24M_f\sqrt{m}\beta & o.w. \end{cases}$$

Interestingly, Theorem 2 holds without making strong convexity assumption on the individual functions, but only requires that the adaptive strong convexity parameter $\mu_\star$ to be positive, as, otherwise, the result is vacuous.

3.1. Practical Implementation

Towards large scale application of AMOOO with modern deep learning architectures we simplify its implementation. First, we optimize over the simplex as oppose to over $\Delta_{m,\min}$. We conjecture this is a by product of our analysis. In addition, we approximate the Hessian matrices with their diagonal. Prior works used the diagonal Hessian approximation as pre-conditioner [1, 5, 23, 30, 35]. Notably, with this approximation the computational cost of AMOOO scales linearly with number of parameters in the Hessian calculation, instead of quadratically. The following result establishes that the value of optimal curvature, and, hence the convergence rate of AMOOO-GD, degrades continuously with the quality of approximation.

Proposition 3 *Assume that for all* $i \in [m]$ *and* $\mathbf{x} \in \mathcal{X}$ $||\nabla^2 f_i(\mathbf{x}) - \text{Diag}\left(\nabla^2 f_i(\mathbf{x})\right)||_2 \leq \Delta$ *where* $\|\mathbf{A}\|_2$ *is the spectral norm of* $\mathbf{A} \in \mathbb{R}^{d\times d}$. *Let* $\widehat{\mathbf{w}} \in \arg\max_{\mathbf{w}\in\Delta_m} \lambda_{\min}\left(\sum_i w_i \nabla^2 \text{Diag}\left(f_i(\mathbf{x})\right)\right)$. *Then,* $\lambda_{\min}\left(\sum_i \widehat{w}_i \nabla^2 f_i(\mathbf{x})\right) \geq \mu_\star - 2\Delta$.

Next we provide high-level details of our implementation (also see Appendix C).

Diagonal Hessian estimation via Hutchinson's Method. We use the Hutchinson method [5, 13, 35] which provides an estimate to the diagonal Hessian by averaging products of the Hessian with random vectors. Importantly, the computational cost of the Hutchinson method scales linearly with number of parameters.

Maximizing the minimal eigenvalue. Maximizing the minimal eigenvalue of symmetric matrices is known to be a convex problem (Boyd and Vandenberghe [3], Example 3.10) and can be solved via semidefinite programming. For diagonal matrices the problem can be cast as a simpler max-min bilinear problem, and, specifically, as $\arg\max_{\mathbf{w}\in\Delta^m} \min_{\mathbf{q}\in\Delta^d} \mathbf{w}^T\mathbf{A}\mathbf{q}$, where d is the dimension of parameters, $\mathbf{A} \in \mathbb{R}^{m\times d}$ and its i^{th} row is the diagonal Hessian of the i^{th} objective, namely, $\forall i \in [m],\ \mathbf{A}[i,:] = \text{diag}(\nabla^2 f_i(\mathbf{x}))$.

This bilinear optimization problem has been well studied in the past [9, 24, 29]. We implemented the PU method of Cen et al. [4] which, loosely speaking, performs iterative updates via exponential gradient descent/ascent. Note that, PU has a closed form update ruke and its computational cost scales linearly with number of parameters.

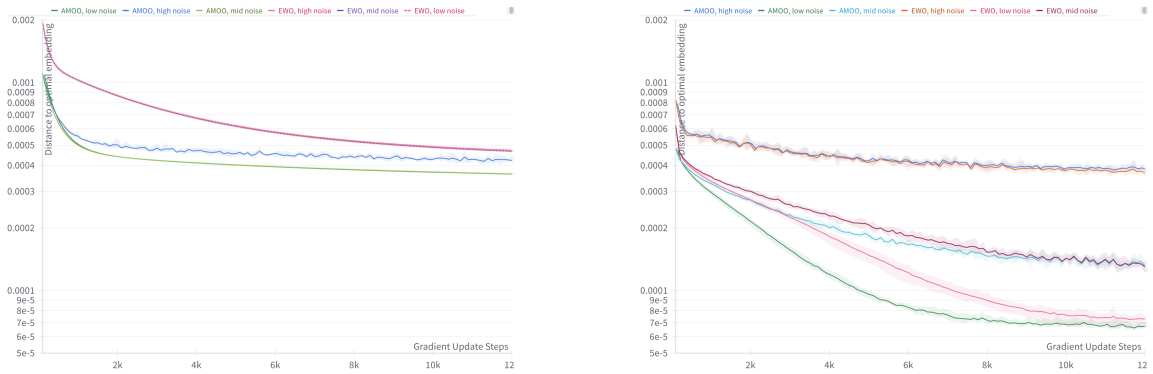

Figure 1: AMOO tested against equal weighting of loss functions (EWO) when optimized by SGD **(left)** or Adam **(right)**. Additionally, we test the effect of additive Normal noise of the optimal representation $h_\theta(x)$. AMOO performs better than its counterpart in all 6 settings.

4. Experiment

We will compare our implementation of AMOOO to a weighting mechanism that equally weighting the objectives (EWO). Specifically, we choose 10 axis-aligned quadratic losses of the form

$$f_i(\mathbf{x}) = (h_\theta(\mathbf{x}) - h_{\theta_\star}(\mathbf{x}))^\top \mathbf{H}_i (h_\theta(\mathbf{x}) - h_{\theta_\star}(\mathbf{x})), \quad \forall i \in [10], \tag{3}$$

where $\mathbf{H}_i \in \mathbb{R}^{10\times 10}$ is a diagonal positive semi-definite Hessian matrix. Both $h_{\theta_\star} : \mathbb{R}^d \to \mathbb{R}^d$ and $h_\theta : \mathbb{R}^d \to \mathbb{R}^d$ are 2-layer neural networks with parameters $\theta_\star$ and θ. Observe that all of the loss functions are minimized when $h_\theta(\mathbf{x}) = h_{\theta_\star}(\mathbf{x})$, and, hence, it is an instance of the AMOO setting.

In our experiment, we choose all but one of the losses to have low curvature, **simulating a selection example** (see Section 2). The features $\mathbf{x}$ are generated by sampling from a d dimensional Normal distribution $\mathcal{N}(0, \mathbf{I}_{10})$, and the targets are perturbed by an additional Normal noise, namely, $\mathbf{y} = h_{\theta_\star}(\mathbf{x}) + \epsilon_\sigma$ where $\epsilon_\sigma \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_{10})$, where $\mathbf{I}_d$ is the identity matrix in dimension d. We experiment with three different noise levels by modifying σ. We test both AMOOO and EWO as the mechanisms for calculating a weighted loss $f_\mathbf{w}$ at each iteration, and apply either SGD or Adam optimizer to $f_\mathbf{w}$. In both cases we perform a grid search on the learning rate to find the best performing learning rate parameter. In Figure 1, we show the results of our simulation. Generally, AMOOO performs better than EWO in all settings across optimizers and noise levels. Adam (right) approaches a more optimal representation than SGD. See additional details in Appendix C.

5. Conclusion

In this work, we introduced the AMOO framework to study how aligned multi-objective feedback can improve gradient descent convergence. We designed the AMOOO algorithm, which adaptively weights objectives and offers provably improved convergence guarantees. Our experimental results demonstrate AMOOO's effectiveness optimizing a large number of tasks that share optimal solution. Future research directions include determining optimal rates for AMOO and conducting comprehensive empirical studies. Such advancements will improve our ability to scale learning algorithms to handle a large number of related tasks efficiently.

References

[1] Idan Achituve, Idit Diamant, Arnon Netzer, Gal Chechik, and Ethan Fetaya. Bayesian uncertainty for gradient aggregation in multi-task learning. *arXiv preprint arXiv:2402.04005*, 2024.

[2] Hamsa Bastani. Predicting with proxies: Transfer learning in high dimension. *Management Science*, 67(5):2964–2984, 2021.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] Shicong Cen, Yuting Wei, and Yuejie Chi. Fast policy extragradient methods for competitive games with entropy regularization. *Advances in Neural Information Processing Systems*, 34: 27952–27964, 2021.

[5] Olivier Chapelle, Dumitru Erhan, et al. Improved preconditioner for hessian free optimization. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, volume 201. Citeseer, 2011.

[6] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR, 2018.

[7] Sumanth Chennupati, Ganesh Sistu, Senthil Yogamani, and Samir A Rawashdeh. Multinet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019.

[8] Christoph Dann, Yishay Mansour, and Mehryar Mohri. Reinforcement learning can be more efficient with multiple rewards. In *International Conference on Machine Learning*, pages 6948–6967. PMLR, 2023.

[9] Constantinos Daskalakis and Ioannis Panageas. Last-iterate convergence: Zero-sum games and constrained min-max optimization. *arXiv preprint arXiv:1807.04252*, 2018.

[10] Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

[11] Daria Dzyabura, Srikanth Jagabathula, and Eitan Muller. Accounting for discrepancies between online and offline product evaluations. *Marketing Science*, 38(1):88–106, 2019.

[12] Alexander IJ Forrester, András Sóbester, and Andy J Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the royal society a: mathematical, physical and engineering sciences*, 463(2088):3251–3269, 2007.

[13] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

[14] Kirthevasan Kandasamy, Gautam Dasarathy, Junier B Oliva, Jeff Schneider, and Barnabás Póczos. Gaussian process bandit optimisation with multi-fidelity evaluations. *Advances in neural information processing systems*, 29, 2016.

[15] Kirthevasan Kandasamy, Gautam Dasarathy, Barnabas Poczos, and Jeff Schneider. The multi-fidelity multi-armed bandit. *Advances in neural information processing systems*, 29, 2016.

[16] Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity bayesian optimisation with continuous approximations. In *International conference on machine learning*, pages 1799–1808. PMLR, 2017.

[17] Seung Hyun Lee, Yinxiao Li, Junjie Ke, Innfarn Yoo, Han Zhang, Jiahui Yu, Qifei Wang, Fei Deng, Glenn Entis, Junfeng He, et al. Parrot: Pareto-optimal multi-reward reinforcement learning framework for text-to-image generation. *arXiv preprint arXiv:2401.05675*, 2024.

[18] Shibo Li, Robert M Kirby, and Shandian Zhe. Deep multi-fidelity active learning of high-dimensional outputs. *arXiv preprint arXiv:2012.00901*, 2020.

[19] Shibo Li, Jeff M Phillips, Xin Yu, Robert Kirby, and Shandian Zhe. Batch multi-fidelity active learning with budget constraints. *Advances in Neural Information Processing Systems*, 35: 995–1007, 2022.

[20] Baijiong Lin and Yu Zhang. Libmtl: A python library for deep multi-task learning. *The Journal of Machine Learning Research*, 24(1):9999–10005, 2023.

[21] Baijiong Lin, Feiyang Ye, Yu Zhang, and Ivor W Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *arXiv preprint arXiv:2111.10603*, 2021.

[22] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34: 18878–18890, 2021.

[23] Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. *arXiv preprint arXiv:2305.14342*, 2023.

[24] Panayotis Mertikopoulos, Houssam Zenati, Bruno Lecouat, Chuan-Sheng Foo, Vijay Chandrasekhar, and Georgios Piliouras. Mirror descent in saddle-point problems: Going the extra (gradient) mile. *arXiv preprint arXiv:1807.02629*, 2018.

[25] Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. *arXiv preprint arXiv:2202.01017*, 2022.

[26] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[27] Yurii Nesterov et al. *Lectures on convex optimization*, volume 137. Springer, 2018.

[28] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[29] Sasha Rakhlin and Karthik Sridharan. Optimization, learning, and games with predictable sequences. *Advances in Neural Information Processing Systems*, 26, 2013.

[30] Tom Schaul, Sixin Zhang, and Yann LeCun. No more pesky learning rates. In *International conference on machine learning*, pages 343–351. PMLR, 2013.

[31] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

[32] Jialin Song, Yuxin Chen, and Yisong Yue. A general framework for multi-fidelity bayesian optimization with gaussian processes. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3158–3167. PMLR, 2019.

[33] Shion Takeno, Hitoshi Fukuoka, Yuhki Tsukada, Toshiyuki Koyama, Motoki Shiga, Ichiro Takeuchi, and Masayuki Karasuyama. Multi-fidelity bayesian optimization with max-value entropy search and its parallelization. In *International Conference on Machine Learning*, pages 9334–9345. PMLR, 2020.

[34] Jian Wu, Saul Toscano-Palmerin, Peter I Frazier, and Andrew Gordon Wilson. Practical multi-fidelity bayesian optimization for hyperparameter tuning. In *Uncertainty in Artificial Intelligence*, pages 788–798. PMLR, 2020.

[35] Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney. Adahessian: An adaptive second order optimizer for machine learning. In *proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10665–10673, 2021.

[36] Jiaxiang Yi, Fangliang Wu, Qi Zhou, Yuansheng Cheng, Hao Ling, and Jun Liu. An active-learning method based on multi-fidelity kriging model for structural reliability analysis. *Structural and Multidisciplinary Optimization*, 63:173–195, 2021.

[37] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.

[38] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pages 737–744. IEEE, 2020.

Appendix A. Related Work

A.1. Multi-task Learning and Gradient Weights

Our work is closely related optimization methods from the multi-task learning (MTL) literature, particularly those that integrate weights into the task gradients or losses. The *multiple gradient*

descent algorithm (MGDA) approach of [10], which proposes an optimization objective that gives rise to a weight vector that implies a descent direction for all tasks. MGDA converges to a point on the Pareto set. MGDA was introduced into the deep MTL setting in [31], which propose extensions to MGDA weight calculation that can be more efficiently solved.

The PCGrad paper [37] identified that conflicting gradients, especially when there is high positive curvature and differing gradient magnitudes, can be detrimental to MTL. The authors then propose to alter the gradients to remove this conflict (by projecting each task's gradient to the normal plane of another task), forming the basis for the PCGrad algorithm. Another work that tackles conflicting gradients is the *conflict-averse gradient descent* (CAGrad) method of [22]. CAGrad generalizes MGDA: its main idea is to minimize a notion of "conflict" between gradients from different tasks, while staying nearby the gradient of the average loss. Notably, CAGrad maintains convergence toward a minimum of the average loss. Another way to handle gradient conflicts is the Nash-MTL method of [25], where the gradients are combined using a bargaining game. Other optimization techniques for MTL include tuning gradient magnitudes so that all tasks train at a similar rate [6], taking the geometric mean of task losses [7], and random weighting [21].

Our approach, `AMOOO`, is similar to existing work in that it also computes gradient weights in order to combine information from multiple pieces of feedback. However, unlike previous work, we focus on exploiting prior knowledge that the objectives are *aligned* and show both theoretically and empirically that such knowledge can be beneficial for optimization.

A.2. Proxy, Multi-fidelity, and Sim-to-real Optimization

Two other streams of related work are (1) machine learning using proxies and (2) multi-fidelity optimization. These works stand out from MTL in that they both focus on using *closely related* objectives, while traditional MTL typically considers a set of tasks that are more varied in nature. Proxy-based machine learning attempts to approximate the solution of a primary "gold" task (for which data is expensive or sparsely available) by making use of a proxy task where data is more abundant [2, 11].

Similarly, multi-fidelity optimization makes use of data sources of varying levels of accuracy (and potentially lower computational cost) to optimize a target objective [12]. In particular, the idea of using multiple closely-related tasks of varying levels of fidelity has seen adoption in settings where function evaluations are expensive, including bandits [14, 15], Bayesian optimization [16, 32–34], and active learning [18, 19, 36]. Sim-to-real learning can be thought of as a particular instance of multi-fidelity optimization, where one hopes to learn real world behavior via simulations (typically in robotics) [28, 38]. In many of these papers, however, the objectives are queried one at a time, differing slightly from the MTL or AMOO settings.

The motivations behind the AMOO setting are clearly similar to those of proxy optimization, multi-fidelity optimization, and sim-to-real learning. However, our papers takes a pure optimization and gradient-descent perspective, which to our knowledge, is novel in the literature.

Appendix B. Proofs of Theoretical Results

B.1. Assumptions & Consequences

In this section we formally provide our working assumptions. We assume access to multi-objective feedback with m objectives $F(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))$. Considering AMOO, we assume the functions are aligned in the sense of equation 1, namely, that they share an optimal solution.

We assume that the exist a local weighting for which the the minimal eigenvalue of the Hessian of the weighted function is at least $\mu_\star$. Further, we define the following quantities, for the single and multi optimization settings:

$$\|\mathbf{y}\|_{\mathbf{x}}^2 := \|\mathbf{y}\|_{\nabla^2 f(\mathbf{x})}$$
$$\|\mathbf{y}\|_{\mathbf{x},\mathbf{w}}^2 := \|\mathbf{y}\|_{\nabla^2 f_{\mathbf{w}}(\mathbf{x})} .$$

Assumption 4 (Smoothness) *All function are β-smooth. $\forall i \in [m]$, $f_i : \mathbb{R}^n \to \mathbb{R}$ it holds that $\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}$:*

$$f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^\top (\mathbf{y} - \mathbf{x}) + \frac{\beta}{2} \|\mathbf{x} - \mathbf{y}\|^2 .$$

Assumption 5 (Self-concordant) *All functions are self-concordant with $M_f \geq 0$ parameter. $\forall i \in [m]$ $f : \mathbb{R}^n \to \mathbb{R}$ and $\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}$:*

$$\langle \nabla^3 f_i(\mathbf{x})[\mathbf{y}]\mathbf{y}, \mathbf{y} \rangle \preceq 2M_f \|\mathbf{y}\|_{\mathbf{x}}^3 ,$$

where $\nabla^3 g(\mathbf{x})[\mathbf{y}] := \lim_{\alpha \to 0} \frac{1}{\alpha} \left(\frac{\nabla^2 g(\mathbf{x}+\alpha\mathbf{y}) - \nabla^2 g(\mathbf{x})}{\alpha} \right)$ is the directional derivative of the hessian in $\mathbf{y}$.

These assumptions have the following important consequences.

Lemma 6 (Theorem 5.1.8 & Lemma 5.1.5, Nesterov [26]) *Let $f : \mathcal{X} \to \mathbb{R}$ be an M_f self-concordant function. Let $x, y \in \mathcal{X}$, we have*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{1}{M_f^2} \omega \left(M_f \|\mathbf{y} - \mathbf{x}\|_{\mathbf{x}} \right) ,$$

where $\omega(t) = t - \ln(1 - t)$, and, for any $t > 0$, $\omega(t) \geq \frac{t^2}{2(1+t)}$.

Lemma 7 (Theorem 5.1.1, Nesterov et al. [27]) *Let $f_1, f_2 : \mathcal{X} \to \mathbb{R}$ be M_f self-concordant functions. Let $w_1, w_2 > 0$. Then, $f = w_1 f_1 + w_2 f_2$ is $M = \max_i \{\frac{1}{\sqrt{w_i}}\} M_f$ self-concordant function.*

Lemma 8 *Let $\{f_i : \mathcal{X} \to \mathbb{R}\}_{i=1}^n$ be M_f self-concordant functions. Let $\{w_i > 0\}$. Then, $f = \sum_{i=1}^n w_i f_i$ is $M = \max_i \{\frac{1}{\sqrt{w_i}}\} M_f$ self-concordant function.*

Proof Let $f = \sum_{i=1}^{n} w_i f_i$. We prove it by using induction.
Basis: $n = 2$. Using Lemma 7 we obtain that f is $\max_{i\in[1,2]}\{\frac{1}{\sqrt{w_i}}\}M_f$ self-concordant function.
Induction assumption: For every $n < k$ it hold that f is $\max_{i\in[1,n]}\{\frac{1}{\sqrt{w_i}}\}M_f$ self-concordant function.
Induction step: Let $f = \sum_{i=1}^{k} w_i f_i$. Define $g = \sum_{i=1}^{k-1} w_i f_i$. From the *Induction assumption* it hold that g is $\max_{i\in[1,k-1]}\{\frac{1}{\sqrt{w_i}}\}M_f$ self-concordant function. Since $f = g + w_k f_k$, by using Lemma 7 we obtain that f is $\max\{\max_{i\in[1,k-1]}\{\frac{1}{\sqrt{w_i}}\}, \frac{1}{\sqrt{w_k}}\}M_f = \max_{i\in[1,k]}\{\frac{1}{\sqrt{w_i}}\}M_f$ self-concordant function. ■

Lemma 9 (Standard, E.g., 9.17 Boyd and Vandenberghe [3]) *Let $f : \mathbb{R}^n \to \mathbb{R}$ a β-smooth over $\mathcal{X}$, and let $\mathbf{x}_\star \in \arg\min_{\mathbf{x}\in\mathbb{R}} f(\mathbf{x})$. Then, it holds that*

$$\|\nabla f(\mathbf{x})\|^2 \leq 2\beta \left(f(\mathbf{x}) - f(\mathbf{x}_\star)\right).$$

Further, we have the following simple consequence of the AMOO setting.

Lemma 10 *For all $\mathbf{w} \in \Delta_m$ and $\mathbf{x} \in \mathcal{X}$ it holds that $f_{\mathbf{w}}(\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x}_\star) \geq 0$.*

Proof
Observe that $f_{\mathbf{w}}(\mathbf{x}) - f_{\mathbf{w}}(\mathbf{x}_\star) = \sum_{i=1}^{m} w_i \left(f_i(\mathbf{x}) - f_i(\mathbf{x}_\star)\right)$. Since $\mathbf{x}_\star$ is the optimal solution for all objectives it holds that $f_i(\mathbf{x}) - f_i(\mathbf{x}_\star) \geq 0$. The lemma follows from the fact $w_i \geq 0$ for all $i \in [m]$. ■

Further, recall that the following observation holds.

Observation 11 *Let $\mathbf{w} \in \Delta_m$. Assume $\{f_i\}_{i=1}^m$ are β smooth. Then, $f_{\mathbf{w}}(\mathbf{x}) := \sum_{i=1}^{m} w_i f_i(\mathbf{x})$ is also β smooth.*

B.2. Proof of Proposition 3

Recall the following results which is a corollary of Weyl's Theorem.

Theorem 12 (Weyl's Theorem) *Let $\mathbf{A}$ and Δ be symmetric matrices in $\mathbb{R}^{d\times d}$. Let $\lambda_j(\mathbf{A})$ be the jth largest eigenvalue of a matrix $\mathbf{A}$. Then, for all $j \in [d]$ it holds that $\|\lambda_j(\mathbf{A}) - \lambda_j(\mathbf{A}+\Delta)\| \leq \|\Delta\|_2$, where $\|\Delta\|_2$ is the operator norm of Δ.*

Proposition 3 is a direct outcome of this result. We establish it for a general deviation in Hessian matrices without requiring it to be necessarily diagonal.
Proof
Denote $\mathbf{A}_i := \nabla^2 f(\mathbf{x}) + \Delta_i$. Let $\mathbf{w}_\star$ denote the solution of,

$$\mathbf{w}_\star \in \arg\max_{\mathbf{w}\in\Delta} \lambda_{\min}\left(\sum_i w_i \nabla^2 f_i(\mathbf{x})\right),$$

and let $g(\mathbf{w}_\star)$ denote the optimal value, $g(\mathbf{w}_\star) = \lambda_{\min}\left(\sum_i w_{\star,i}\nabla^2 f_i(\mathbf{x})\right)$. Similarly, let $\hat{\mathbf{w}}_\star$ denote the solution of the optimization problem of the perturbed problem:

$$\hat{\mathbf{w}}_\star \in \arg\max_{\mathbf{w}\in\Delta} \lambda_{\min}\left(\sum_i w_i \mathbf{A}_i\right),$$

and denote $\hat{g}(\hat{\mathbf{w}}_\star)$ as its value, $\hat{g}(\hat{\mathbf{w}}_\star) = \lambda_{\min}\left(\sum_i \hat{w}_{\star,i}\mathbf{A}_i\right)$. Then, the following holds.

$$\begin{aligned}
g(\mathbf{w}_\star) &= g(\mathbf{w}_\star) - \hat{g}(\mathbf{w}_\star) + \hat{g}(\mathbf{w}_\star) - \hat{g}(\hat{\mathbf{w}}_\star) + \hat{g}(\hat{\mathbf{w}}_\star) - g(\hat{\mathbf{w}}_\star) + g(\hat{\mathbf{w}}_\star) \\
&\overset{(1)}{\leq} g(\mathbf{w}_\star) - \hat{g}(\mathbf{w}_\star) + \hat{g}(\hat{\mathbf{w}}_\star) - g(\hat{\mathbf{w}}_\star) + g(\hat{\mathbf{w}}_\star) \\
&\leq |g(\mathbf{w}_\star) - \hat{g}(\mathbf{w}_\star)| + |\hat{g}(\hat{\mathbf{w}}_\star) - g(\hat{\mathbf{w}}_\star)| + g(\hat{\mathbf{w}}_\star) \\
&\overset{(2)}{\leq} 2\|\Delta\|_2 + g(\hat{\mathbf{w}}_\star).
\end{aligned}$$

(1) holds since $\hat{g}(\mathbf{w}_\star) - \hat{g}(\hat{\mathbf{w}}_\star) \leq 0$ by the optimality of $\hat{\mathbf{w}}_\star$ on $\hat{g}$. Further, (2) holds due to Weyl's Thoerem (Theorem 12) and the assumptions of the approximation error. Recall that for any $\mathbf{w} \in \Delta_m$ it holds that

$$\left\|\sum_i w_i \mathbf{A}_i - \sum_i w_i \nabla^2 f_i(\mathbf{x})\right\|_2 \leq \sum_i w_i \left\|\mathbf{A}_i - \nabla^2 f_i(\mathbf{x})\right\|_2 \leq \|\Delta\|_2$$

since $\sum_i w_i = 1$. Hence, by Weyl's theorem it holds that

$$|g(\mathbf{w}_\star) - \hat{g}(\mathbf{w}_\star)\| \leq \|\Delta\|_2 \text{ and } |g(\hat{\mathbf{w}}_\star) - \hat{g}(\hat{\mathbf{w}}_\star)\| \leq \|\Delta\|_2.$$

Finally, since $g(\mathbf{w}_\star) \geq \mu_\star$, by Definition 1, we get that

$$g(\hat{\mathbf{w}}_\star) \geq \mu_\star - 2\|\Delta\|_2,$$

which concludes the proof. ■

B.3. Proof of Theorem 2

In highlevel, the proof follows the standard convergence analysis of μ-strongly convex and L-smooth function, while applying Lemma 6, instead of using only properties of strongly convex functions alone.

In addition, we choose the minimal weight value, $w_{\min}$, such that the weighted function at each iteration $f_{\mathbf{w}_k}$ has a sufficiently large self-concordant parameter, while the minimal eigenvalue of its Hessian is close to $\mu_\star$. Before proving Theorem 2, we provide two results that allow us to control these two aspects.

Lemma 13 *For any iteration k, the function $f_{\mathbf{w}_k}$ is $1/\sqrt{w_{\min}}M_f$ self-concordant.*

Proof This is a direct consequence of Lemma 8 and the fact Algorithm 2 sets the weights by optimizing over a set where the weight vector, $\mathbf{w}$. is lower bounded by $w_{\min}$. ■

Lemma 14 *For any iteration k, we have $\lambda_{\min}\left(\nabla^2 f_{\mathbf{w}_k}\right) \geq \mu_\star - 2mw_{\min}\beta$.*

Proof

To establish the lemma we want to show that for any $\mathbf{w} \in \Delta_m$ there exists $\widehat{\mathbf{w}} \in \Delta_{m,w_{\min}}$ such that $\lambda_{\min}\left(\sum_i \hat{w}_i \nabla^2 f_i(\mathbf{x}_t)\right) \geq \lambda_{\min}\left(\sum_i w_i \nabla^2 f_i(\mathbf{x}_t)\right) - w_{\min}\beta$. We start by bounding the following term $\left\|\nabla^2 f_{\mathbf{w}}(\mathbf{x}) - \nabla^2 f_{\hat{\mathbf{w}}}(\mathbf{x})\right\|_2$ for any $\mathbf{x} \in \mathcal{X}$. We have

$$\left\|\sum_i (w_i - \hat{w}_i)\nabla^2 f_i(\mathbf{x})\right\|_2 \leq \sum_i |w_i - \hat{w}_i| \left\|\nabla^2 f_i(\mathbf{x})\right\|_2 \leq \beta \sum_i |w_i - \hat{w}_i|,$$

while the last inequality holds since $\{f_i\}_{i\in[m]}$ are β smooth. Since for any $\mathbf{w} \in \Delta_m$ there exist $\hat{\mathbf{w}} \in \Delta_{m,w_{\min}}$ such that $\sum_i |w_i - \hat{w}_i| \leq 2mw_{\min}$, we obtain that for every $\mathbf{x} \in \mathcal{X}$ it holds that

$$\left\|\nabla^2 f_{\mathbf{w}}(\mathbf{x}) - \nabla^2 f_{\hat{\mathbf{w}}}(\mathbf{x})\right\|_2 \leq 2mw_{\min}\beta.$$

Thus, by using Theorem 12 we have

$$|\lambda_{\min}(\nabla^2 f_{\mathbf{w}}(\mathbf{x})) - \lambda_{\min}(\nabla^2 f_{\hat{\mathbf{w}}}(\mathbf{x}))| \leq \left\|\nabla^2 f_{\mathbf{w}}(\mathbf{x}) - \nabla^2 f_{\hat{\mathbf{w}}}(\mathbf{x})\right\|_2 \leq 2mw_{\min}\beta.$$

Recall that $\lambda_{\min}(\nabla^2 f_{\mathbf{w}}(\mathbf{x})) \geq \mu^*$ assuming Definition 1 holds. Then, we obtain

$$\lambda_{\min}(\nabla^2 f_{\hat{\mathbf{w}}}(\mathbf{x})) \geq \mu^* - 2mw_{\min}\beta.$$

■

With these two results we are ready to prove Theorem 2.

Proof

The GD update rule is given by $\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \nabla f_{\mathbf{w}_k}(\mathbf{x}_k)$, where η is the step size, and $\mathbf{w}_k \in \arg\max_{\mathbf{w}\in\Delta_m} \lambda_{\min}\left(\sum_i w_i \nabla^2 f_i(\mathbf{x}_t)\right)$. With the assumption that $\max_{\mathbf{w}\in\Delta_m} \lambda_{\min}\left(\nabla^2 f_{\mathbf{w}_k}(\mathbf{x}_k)\right) = \mu_\star > 0$, Lemma 14, and since we set $w_{\min} = \mu_\star/(8m\beta)$ we have that

$$\lambda_{\min}\left(\nabla^2 f_{\mathbf{w}_k}(\mathbf{x}_k)\right) \geq \mu_\star - 4mw_{\min}\beta := \mu_\star/2 = \widehat{\mu_\star}, \tag{4}$$

for all iterations k.

We bound the squared distance between $\mathbf{x}_{k+1}$ and $\mathbf{x}_\star$:

$$\begin{aligned}\|\mathbf{x}_{k+1} - \mathbf{x}_\star\|^2 &= \|\mathbf{x}_k - \eta\nabla f_{\mathbf{w}_k}(\mathbf{x}_k) - \mathbf{x}_\star\|^2 \\ &= \|\mathbf{x}_k - \mathbf{x}_\star\|^2 - 2\eta\langle\nabla f_{\mathbf{w}_k}(\mathbf{x}_k), \mathbf{x}_k - \mathbf{x}^*\rangle + \eta^2 \|\nabla f_{\mathbf{w}_k}(\mathbf{x}_k)\|^2\end{aligned}$$

By Lemma 13 it holds that $f_{\mathbf{w}_k}$ is

$$\widehat{M_f} := 1/\sqrt{w_{\min}} M_f \leq 3\sqrt{m\beta} M_f/\sqrt{\mu_\star}$$

self concordant. Then, by applying Lemma 6 with $\mathbf{y} = \mathbf{x}_\star$ and $\mathbf{x} = \mathbf{x}_k$ we have

$$\langle\nabla f_{\mathbf{w}_k}(\mathbf{x}_k), \mathbf{x}_k - \mathbf{x}_\star\rangle \geq f_{\mathbf{w}_k}(\mathbf{x}_k) - f_{\mathbf{w}_k}(\mathbf{x}_\star) + \frac{1}{\widehat{M_f}}\omega\left(\widehat{M_f}\|\mathbf{x}_\star - \mathbf{x}_k\|_{\mathbf{x},\mathbf{w}_k}\right).$$

which allows us to bound $\|\mathbf{x}_{k+1}-\mathbf{x}_\star\|^2$ by

$$\begin{aligned}
&\|\mathbf{x}_k-\mathbf{x}_\star\|^2-2\eta\left(f_{\mathbf{w}_k}(\mathbf{x}_k)-f_{\mathbf{w}_k}(\mathbf{x}_\star)+\frac{1}{\widehat{M_f}}\omega\left(\widehat{M_f}\,\|\mathbf{x}_\star-\mathbf{x}_k\|_{x,\mathbf{w}_k}\right)\right)+\eta^2\|\nabla f_{\mathbf{w}_k}(\mathbf{x}_k)\|^2\\
&\overset{(1)}{\leq}\|\mathbf{x}_k-\mathbf{x}_\star\|^2-\frac{2\eta}{\widehat{M_f}^2}\omega\left(\widehat{M_f}\,\|\mathbf{x}_\star-\mathbf{x}_k\|_{\mathbf{x},\mathbf{w}_k}\right)+2\eta\left(2\beta\eta-1\right)\left(f_{\mathbf{w}_k}(\mathbf{x}_k)-f_{\mathbf{w}_k}(\mathbf{x}_\star)\right)\\
&\overset{(2)}{\leq}\|\mathbf{x}_k-\mathbf{x}_\star\|^2-\frac{1}{\beta\widehat{M_f}^2}\omega\left(\widehat{M_f}\,\|\mathbf{x}_\star-\mathbf{x}_k\|_{\mathbf{x},\mathbf{w}_k}\right)\\
&\overset{(3)}{\leq}\|\mathbf{x}_k-\mathbf{x}_\star\|^2-\frac{1}{2\beta}\frac{\|\mathbf{x}_\star-\mathbf{x}_k\|^2_{\mathbf{x},\mathbf{w}_k}}{1+\widehat{M_f}\,\|\mathbf{x}_\star-\mathbf{x}_k\|_{\mathbf{x},\mathbf{w}_k}}\\
&\overset{(4)}{\leq}\|\mathbf{x}_k-\mathbf{x}_\star\|^2-\frac{\widehat{\mu_\star}}{2\beta}\frac{\|\mathbf{x}_\star-\mathbf{x}_k\|^2}{1+\widehat{M_f}\sqrt{\beta}\,\|\mathbf{x}_\star-\mathbf{x}_k\|}
\end{aligned}$$

where (1) is due to Lemma 9, (2) holds by $f_{\mathbf{w}_k}(\mathbf{x}_k)-f_{\mathbf{w}_k}(\mathbf{x}_\star)\geq 0$ (Lemma 10) and $\eta\left(2\beta\eta-1\right)\leq 0$ since $0<\eta\leq 1/2\beta$, (3) is due to the lower bound on $\omega(t)$ from Lemma 6, and (4) follows from equation (4) and since $f_{\mathbf{w}}$ is β smooth for all $\mathbf{w}\in\Delta_m$.

The above recursive equation results in polynomial contraction for large $\|\mathbf{x}_\star-\mathbf{x}_k\|$, and, then exhibits linear convergence. To see this, let $\kappa:=\frac{\widehat{\mu_\star}}{\beta}$, and examine the two limits.

Linear convergence, $\|\mathbf{x}_\star-\mathbf{x}_k\|\leq\delta/\widehat{M_f}\sqrt{\beta},\ \delta\leq 1$. With this assumption we have the following bound on the recursive equation:

$$\|\mathbf{x}_{k+1}-\mathbf{x}_\star\|^2\leq\left(1-\frac{\kappa}{2(1+\delta)}\right)\|\mathbf{x}_k-\mathbf{x}_\star\|^2.$$

By setting $\delta=1$ we get the result. Further, $\|\mathbf{x}_{k+1}-\mathbf{x}_\star\|^2$ contracts monotonically, without exiting the ball $\|\mathbf{x}_\star-\mathbf{x}_k\|\leq\delta/\widehat{M_f}\sqrt{\beta}$, the linear convergence rate approaches $\kappa/2$.

Polynomial convergence, $\|\mathbf{x}_\star-\mathbf{x}_k\|>1/\widehat{M_f}\sqrt{\beta}$. With this assumption we have the following bound:

$$\|\mathbf{x}_{k+1}-\mathbf{x}_\star\|^2\leq\|\mathbf{x}_k-\mathbf{x}_\star\|^2-\frac{\kappa}{4\widehat{M_f}\sqrt{\beta}}\|\mathbf{x}_k-\mathbf{x}_\star\|.$$

This recursive equation decays in a linear rate and have the following closed form upper bound $\|\mathbf{x}_{k+1}-\mathbf{x}_\star\|\leq\|\mathbf{x}_0-\mathbf{x}_\star\|-k\frac{\kappa}{8\widehat{M_f}\sqrt{\beta}}$.

By plugging the values of $\widehat{M_f}$ and $\widehat{\mu}_\star$ we obtain the final result. ■

Appendix C. Practical Implementation

Dataset. We generate 10 dimensional inputs, $\mathbf{x}\in\mathbb{R}^{10}$ from an independent Normal distribution $\mathcal{N}(0,\mathbf{I}_d)$. The target generating network $h_{\theta_\star}$ is randomly generated. The noise on targets is sampled from a Normal distribution $\epsilon_\sigma\sim\mathcal{N}(0,\sigma\mathbf{I}_d)$ and the noise level is either high $\sigma=1$, medium size $\sigma=0.1$ or low $\sigma=0.001$.

Network architecture. We choose the ground truth network and target network to have the same architecture. Both are 2 layer neural networks with 256 hidden dimensions and ReLu activation. The neural network outputs a vector in dimension 10, similar to the input of the network.

Loss functions. We choose $\mathbf{H}_1 = \mathbf{I}_{10}$, and for $i > 1$ $\mathbf{H}_i = \alpha \mathbf{I}_{10} + (1-\alpha)\mathbf{A}$ and $\alpha = 10^{-4}$ where

$$\mathbf{A}_{i,j} = \begin{cases} 1 & i = j = 1 \\ 0 & o.w., \end{cases}$$

namely, $\mathbf{A}$ is a diagonal matrix with value of 1 in the first diagonal index and zero otherwise.

In this problem, the function generated by the $\mathbf{H}_1$ Hessian has the largest minimal eigenvalue and we expect `AMOOO` to choose this function, whereas `EWO` gives equal weight to every loss function.

Training. We optimize learning rates across a grid of candidates and pick the best performing one on training loss $[1e-5, 1e-4, 1e-3, 1e-2]$, $1e-3$ performed best in all settings. We choose a batch size of 1024. We perform each run with 5 different seeds and average their performance.

General parameters for `AMOOO`. We set the number of samples for the Hutchinson method to be $N_{\text{Hutch}} = 100$. Namley, we estimate the Hessian matrices by averaging $N_{\text{Hutch}} = 100$ estimates obtained from the Hutchnison method. Additionally, we use exponential averaging to update the Hessian matrices with $\beta = 0.99$. Further, at each training step we perform a single update of the weights based on the PU update rule of Cen et al. [4] to solve the max-min Bilinear optimization problem (see Section 3.1).

Validation. We measure the L_2 distance between h_θ and $h_{\theta_\star}$ averaged over $1024 \cdot 10^3$ validation points and measured per dimension. This quantity suppose to approximate the quality of the learned model θ which is given by $\mathbb{E}_x \left[\|h_\theta(\mathbf{x}) - h_{\theta_\star}(\mathbf{x})\|^2 \right]$.