Revisiting Scalarization in Multi-Task Learning

Han Zhao

11/18/2024

Joint IFML/MPG Symposium, Simons Institute

(joint work with Yuzheng Hu, Xiaoyuan Zhang and Meitong Liu)

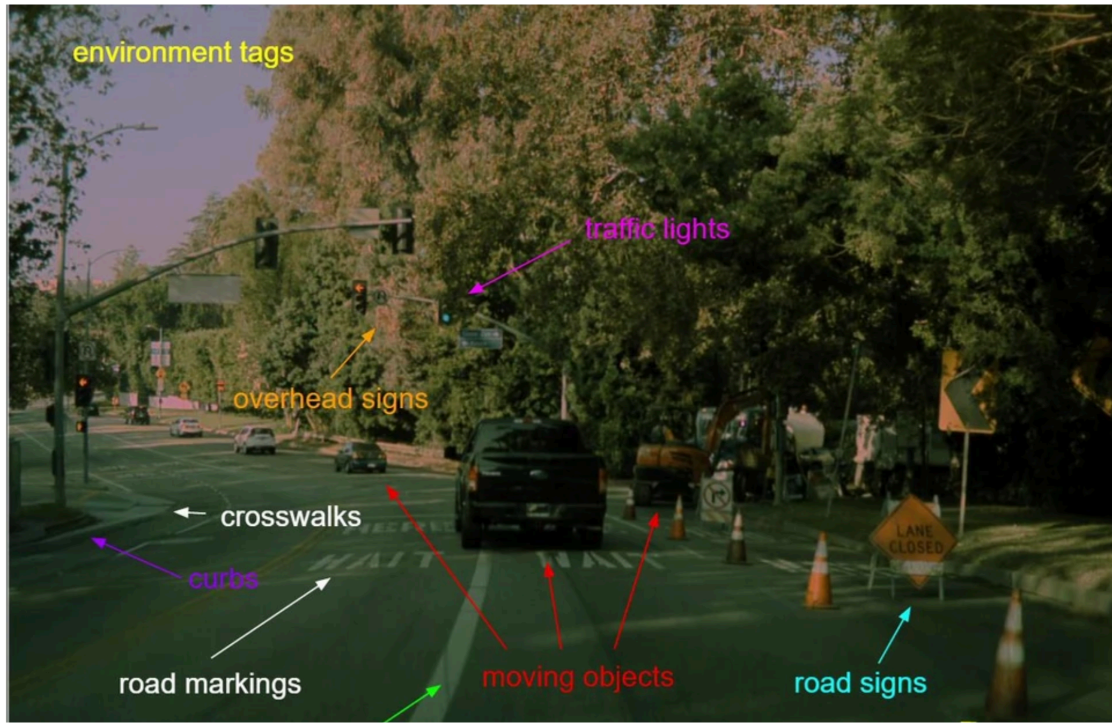
Assistant Professor, Amazon Scholar
hanzhao@illinois.edu
Department of Computer Science
University of Illinois Urbana-Champaign

ILLINOIS
Computer Science
GRAINGER COLLEGE OF ENGINEERING

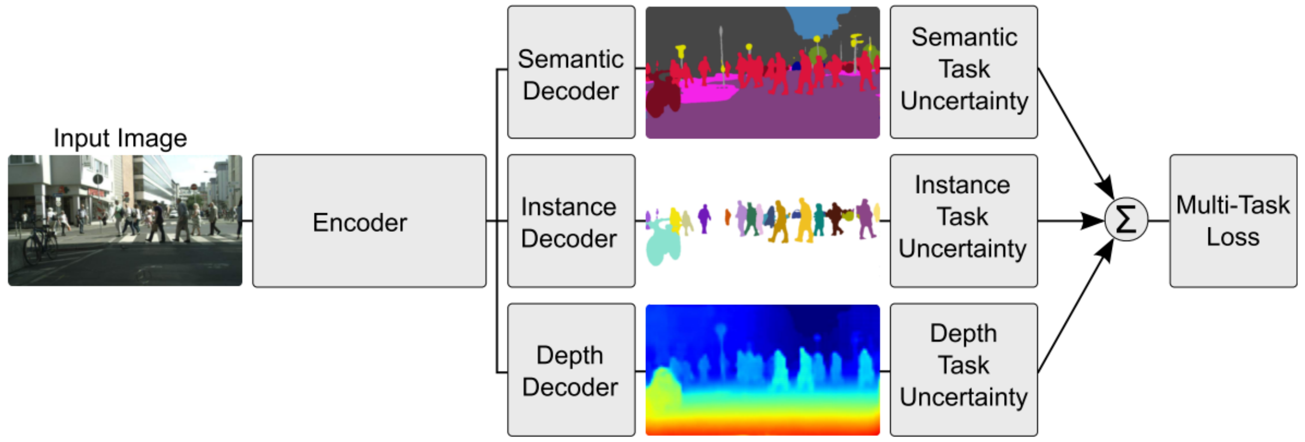
What is Multitask Learning?

Multitask learning:

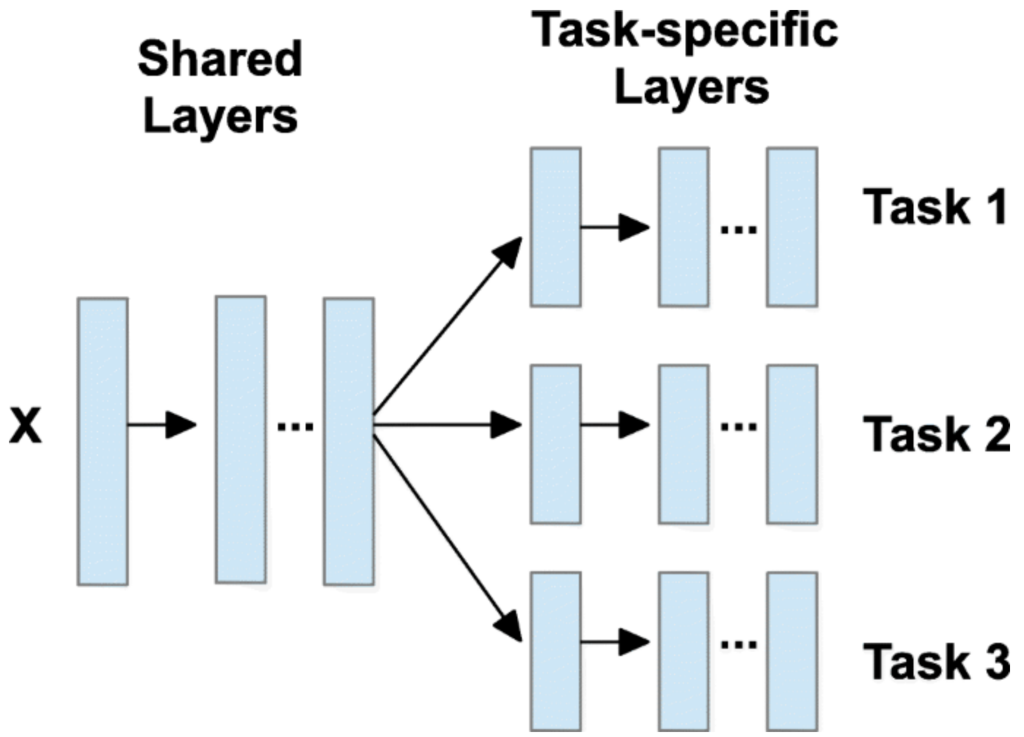
- Suppose we need to model k different tasks



Conceptual Framework



Model Architecture



Multitask Representation Learning

Multitask learning:

- Train a joint model with multiple heads, where one head ~ one task
  - Pros:
    - ➡ Easy to scale to many tasks: adding one more task-specific head
    - ➡ Can exploit potential synergies among tasks
  - Cons:
    - ➡ Hard to design tailored structures
    - ➡ "Negative transfer" could happen due to conflicting tasks

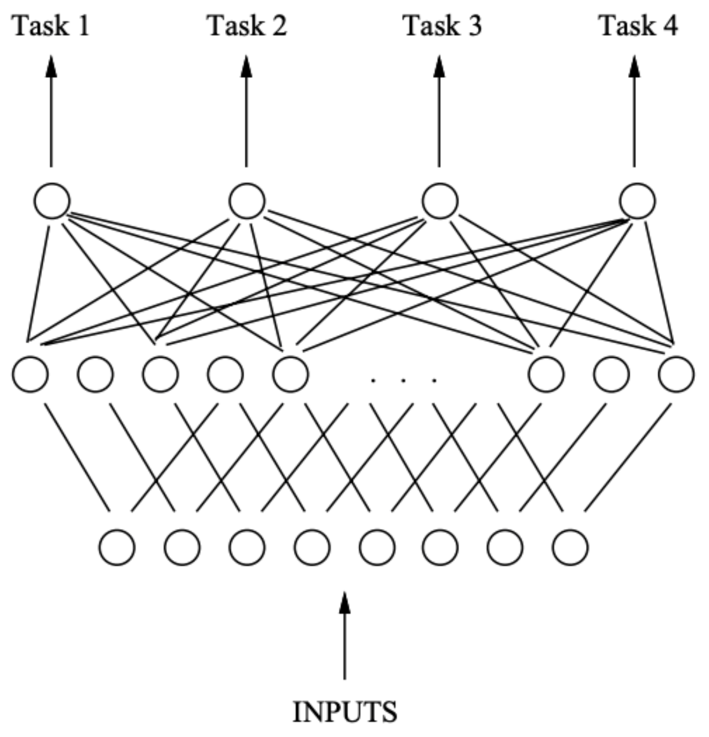


Figure credit: "Multitask Learning", PhD Thesis, Rich Caruana, 1997

Multitask Representation Learning

Multitask learning:

Multi-objective vector loss: $$\ell(\{h_i\}_{i=1}^k, g) := \begin{pmatrix} \mathbb{E}_{\mathcal{D}_1^{\text{tr}}}[\ell_1((h_1 \circ g)(X), Y)] \\ \vdots \\ \mathbb{E}_{\mathcal{D}_k^{\text{tr}}}[\ell_k((h_k \circ g)(X), Y)] \end{pmatrix} \in \mathbb{R}^k$$

Note "$\leq$" is not a total ordering in $\mathbb{R}^k$ for $k > 1$, so in general we pick a preference vector

$$w \in \Delta_{k-1} := \left\{ v \in \mathbb{R}^k : \sum_{i=1}^k v_i = 1, v_i \geq 0 \right\}$$

Linear scalarization

$$\min_{g,h_i} \sum_{i=1}^k \frac{w_i}{n_i} \sum_{j=1}^{n_i} \ell_i((\boxed{h_i} \circ \boxed{g})(x_j^{(i)}), y_j^{(i)})$$

"task-specific header"

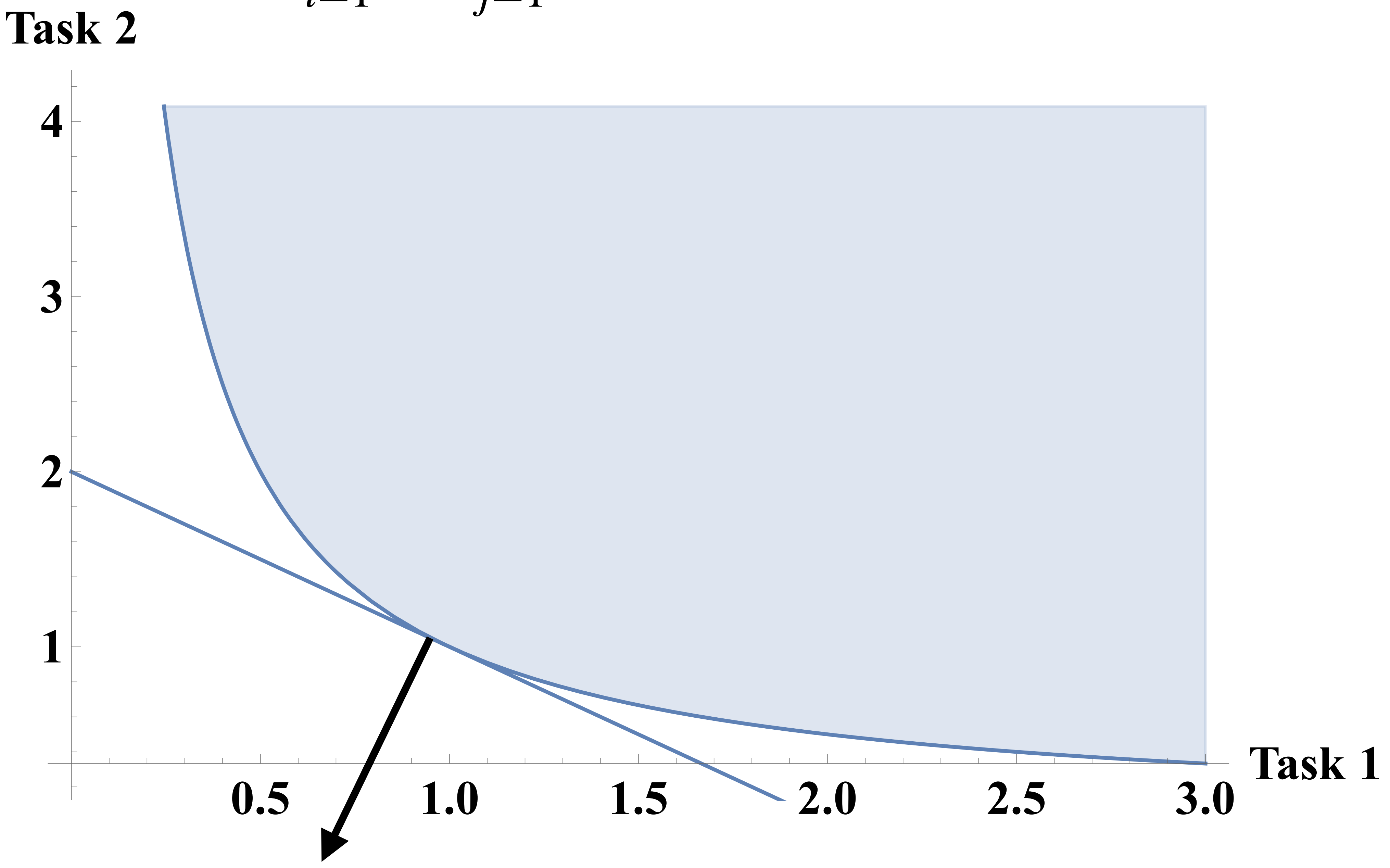
"shared multitask feature learning"

Multitask Representation Learning

Linear scalarization:

MTL Loss: $$\min_{g,h_i} \sum_{i=1}^{k} \frac{w_i}{n_i} \sum_{j=1}^{n_i} \ell_i((h_i \circ g)(x_j^{(i)}), y_j^{(i)})$$

Geometrically:



$$w = (w_1, \ldots, w_j) \in \Delta_{k-1}$$

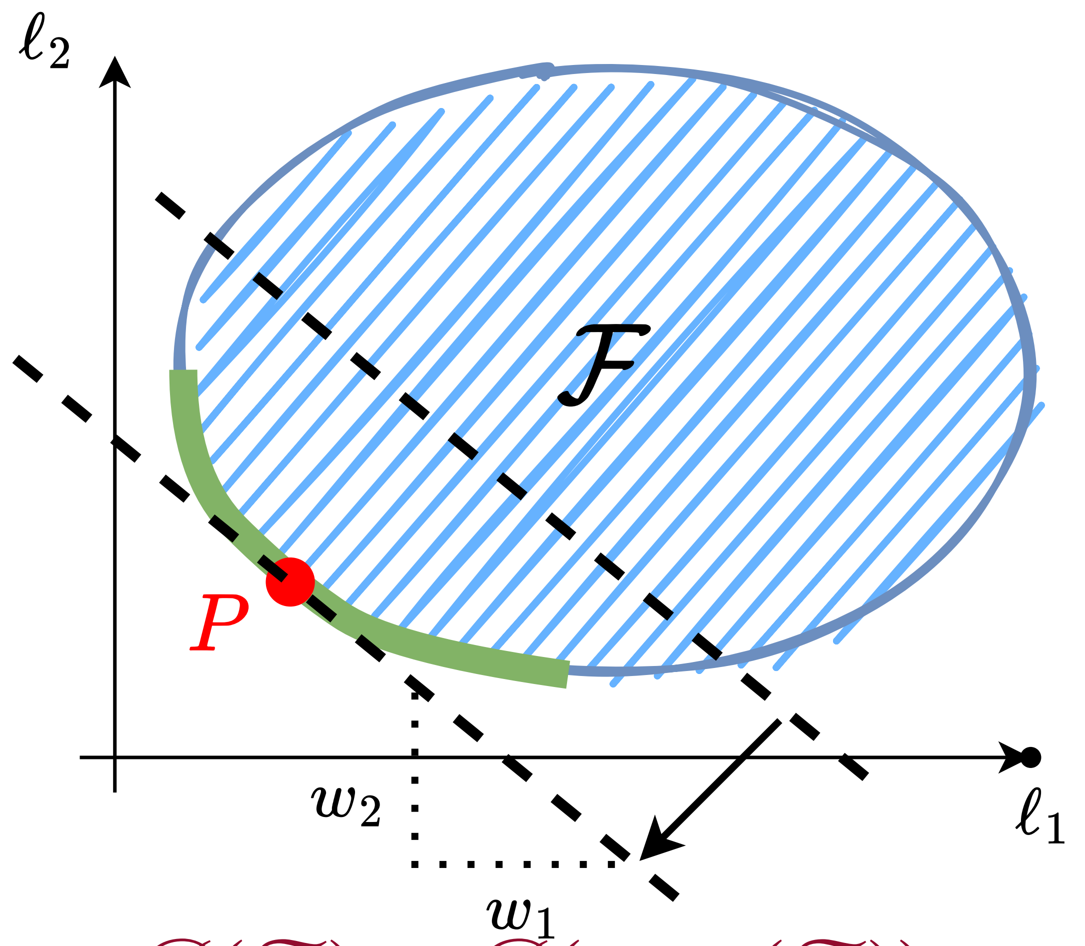
Linear Scalarization for MTL

Linear scalarization:

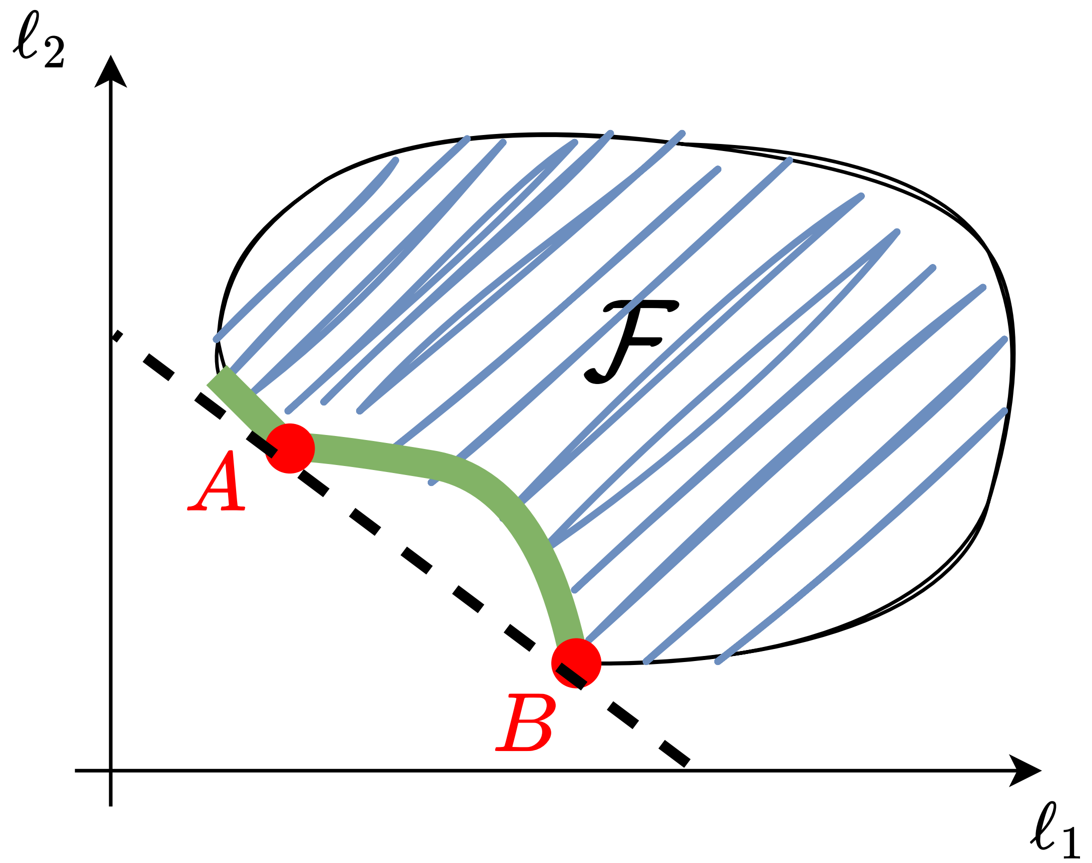
MTL Loss: $$\min_{g,h_i} \sum_{i=1}^{k} \frac{w_i}{n_i} \sum_{j=1}^{n_i} \ell_i((h_i \circ g)(x_j^{(i)}), y_j^{(i)})$$

What's the potential problem of simple scalarization?

Not necessarily possible to strike the right balance among multiple tasks



$$\mathscr{P}(\mathscr{F}) = \mathscr{P}(\mathrm{conv}(\mathscr{F}))$$

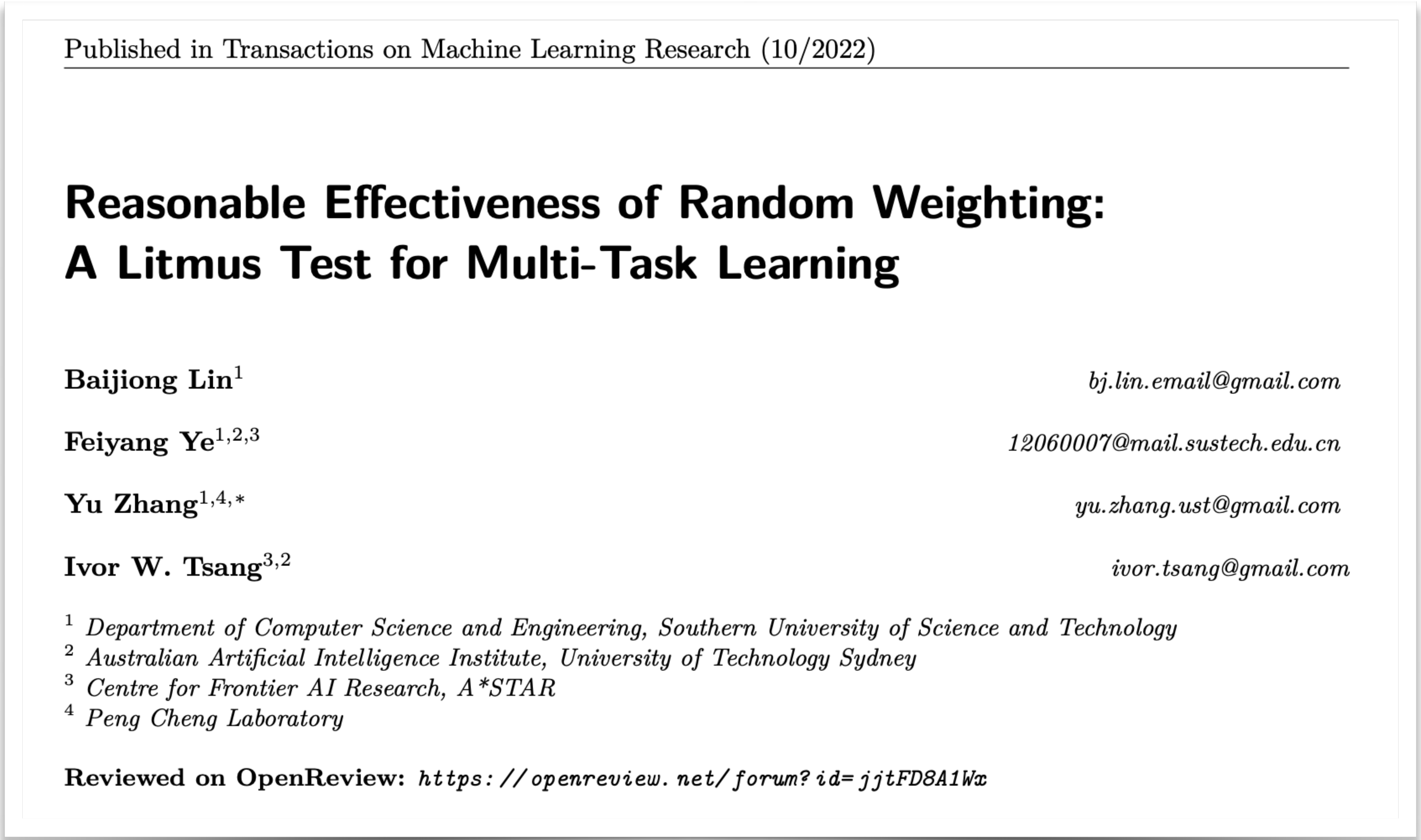


$$\mathscr{P}(\mathscr{F}) \prec \mathscr{P}(\mathrm{conv}(\mathscr{F}))$$

Multi-Objective Optimization

What if we use random weights $w^{(t)} \in \Delta_{k-1}$ at each iteration $t \in [T]$?

A simple starter: simply randomize the combination weighting vector



Published in Transactions on Machine Learning Research (10/2022)

Reasonable Effectiveness of Random Weighting: A Litmus Test for Multi-Task Learning

Baijiong Lin[1] *bj.lin.email@gmail.com*

Feiyang Ye[1,2,3] *12060007@mail.sustech.edu.cn*

Yu Zhang[1,4,*] *yu.zhang.ust@gmail.com*

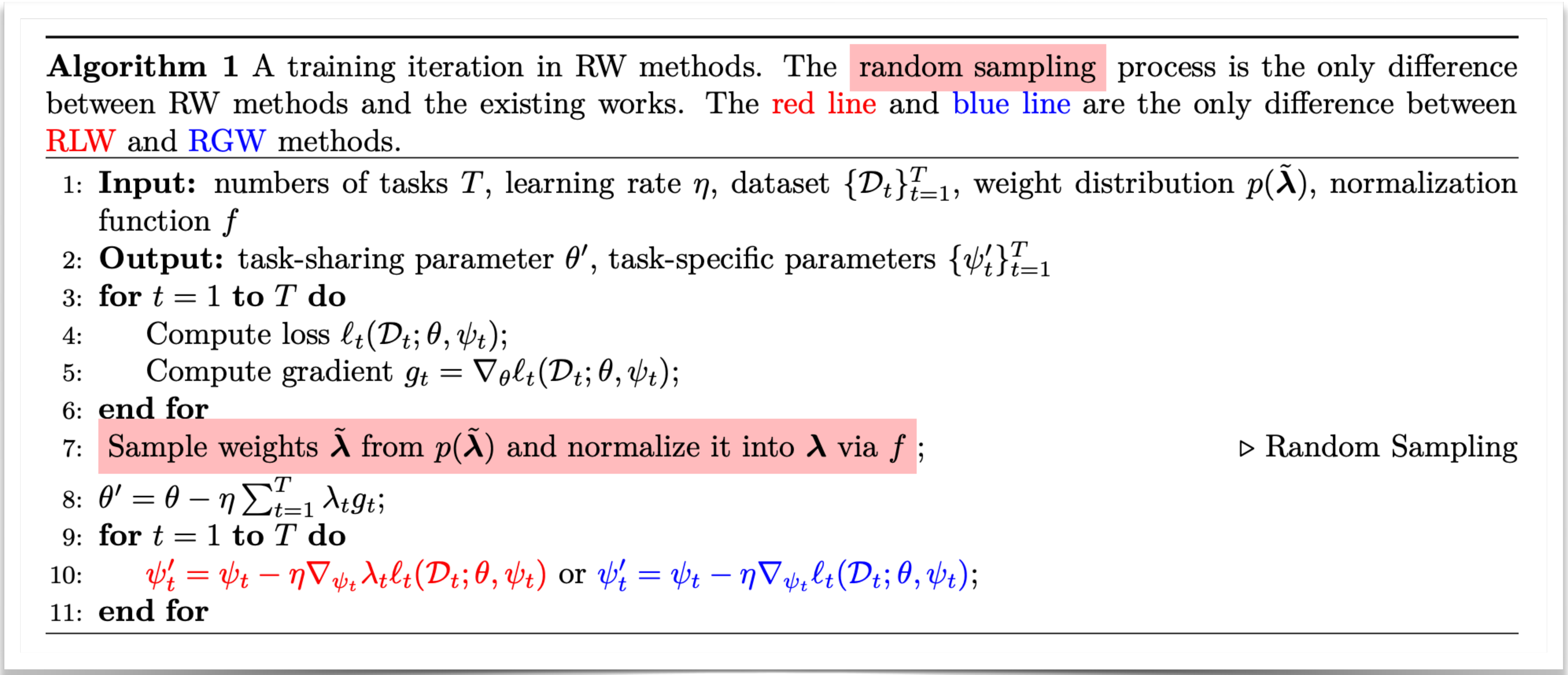
Ivor W. Tsang[3,2] *ivor.tsang@gmail.com*

[1] *Department of Computer Science and Engineering, Southern University of Science and Technology*
[2] *Australian Artificial Intelligence Institute, University of Technology Sydney*
[3] *Centre for Frontier AI Research, A*STAR*
[4] *Peng Cheng Laboratory*

Reviewed on OpenReview: `https://openreview.net/forum?id=jjtFD8A1Wx`

Multi-Objective Optimization

Pseudo-code:



Algorithm 1 A training iteration in RW methods. The random sampling process is the only difference between RW methods and the existing works. The red line and blue line are the only difference between RLW and RGW methods.

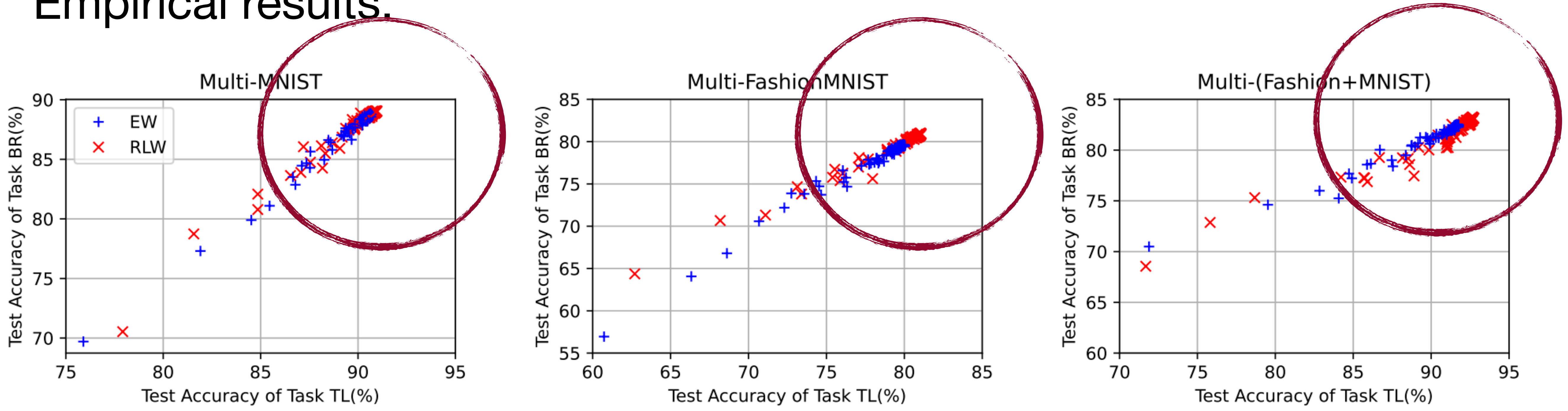
```
1: Input: numbers of tasks $T$, learning rate $\eta$, dataset $\{\mathcal{D}_t\}_{t=1}^T$, weight distribution $p(\tilde{\boldsymbol{\lambda}})$, normalization
   function $f$
2: Output: task-sharing parameter $\theta'$, task-specific parameters $\{\psi'_t\}_{t=1}^T$
3: for $t = 1$ to $T$ do
4:     Compute loss $\ell_t(\mathcal{D}_t; \theta, \psi_t)$;
5:     Compute gradient $g_t = \nabla_\theta \ell_t(\mathcal{D}_t; \theta, \psi_t)$;
6: end for
7: Sample weights $\tilde{\boldsymbol{\lambda}}$ from $p(\tilde{\boldsymbol{\lambda}})$ and normalize it into $\boldsymbol{\lambda}$ via $f$ ;        ▷ Random Sampling
8: $\theta' = \theta - \eta \sum_{t=1}^T \lambda_t g_t$;
9: for $t = 1$ to $T$ do
10:     $\psi'_t = \psi_t - \eta \nabla_{\psi_t} \lambda_t \ell_t(\mathcal{D}_t; \theta, \psi_t)$ or $\psi'_t = \psi_t - \eta \nabla_{\psi_t} \ell_t(\mathcal{D}_t; \theta, \psi_t)$;
11: end for
```

Potential distributions to sample from:

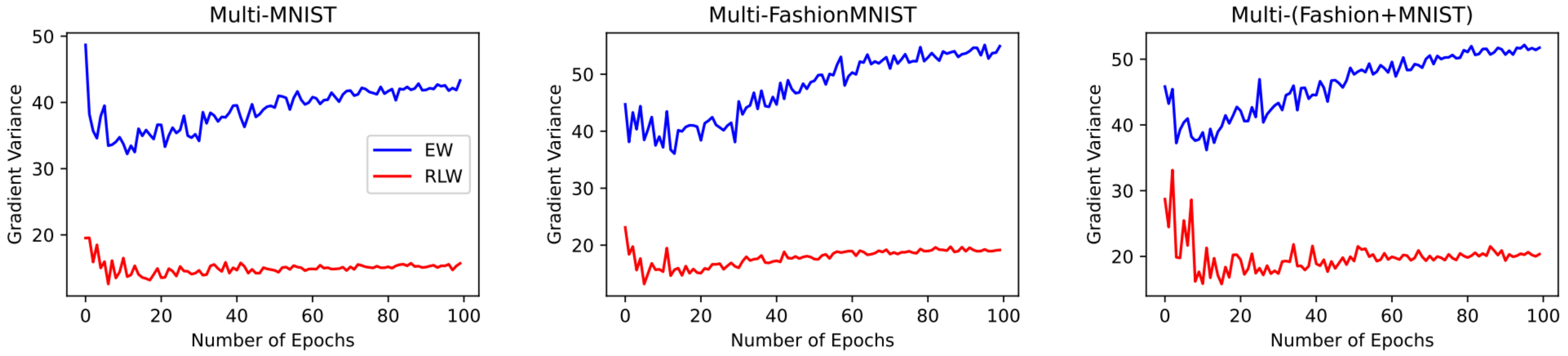
- Uniform, (truncated) Normal, Dirichlet, etc

Multi-Objective Optimization

Empirical results:



Variance of gradient norms:



- EW: equal weighting
- RLW: randomized loss weighting

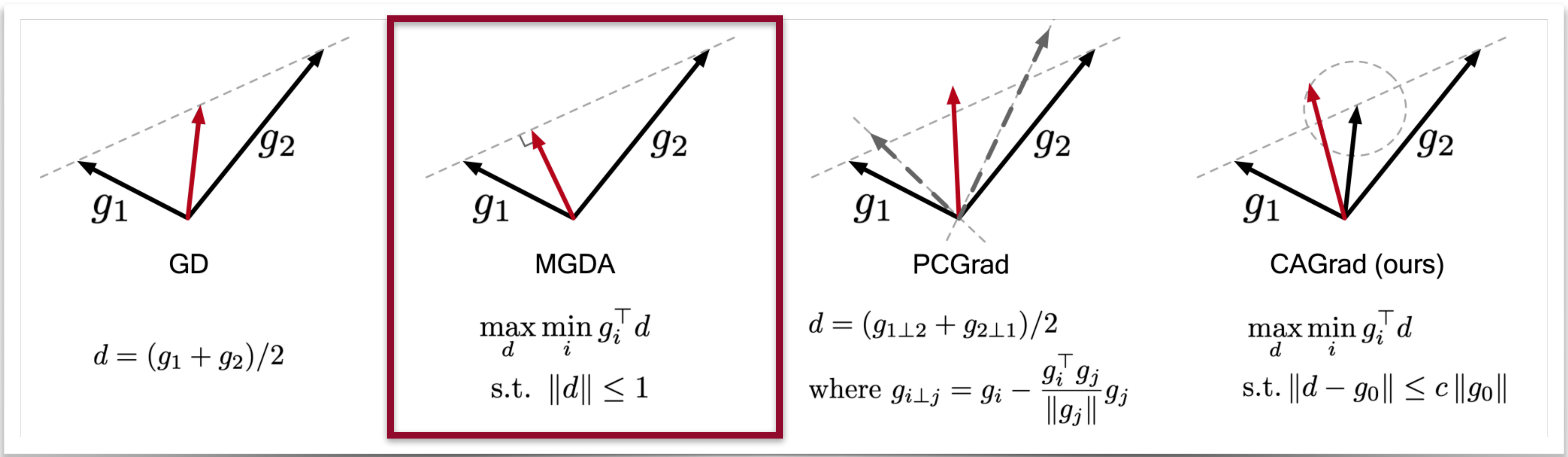
Multi-Objective Optimization

However,

- Unclear this RW (randomized weight) will always converge
- Will it always converge to a Pareto optimal point?

Any other more principled methods to explore the Pareto front?

Yes, for batch learning!



$d = (g_1 + g_2)/2$

$\max_d \min_i g_i^\top d$ s.t. $\|d\| \leq 1$

$d = (g_{1\perp 2} + g_{2\perp 1})/2$ where $g_{i\perp j} = g_i - \frac{g_i^\top g_j}{\|g_j\|} g_j$

$\max_d \min_i g_i^\top d$ s.t. $\|d - g_0\| \leq c\,\|g_0\|$

"Multiple-Gradient Descent Algorithm (MGDA) for Multi-Objective Optimization", Comptes Rendus Mathématique, Désidéri, 2012.

"Multi-Task Learning as Multi-Objective Optimization", NeurIPS'18, Sener and Kolton.

"Gradient Surgery for Multi-Task Learning", NeurIPS'20, Yu et al.

"Conflict-Averse Gradient Descent for Multi-task Learning", NeurIPS'21, Liu et al.

Multi-Objective Optimization

Multiple-Gradient Descent Algorithm

Let $g_i, i \in [k]$ be the gradient for the k tasks at a certain iteration

First order improvement along direction d: $g_i^\top d$

Primal problem:

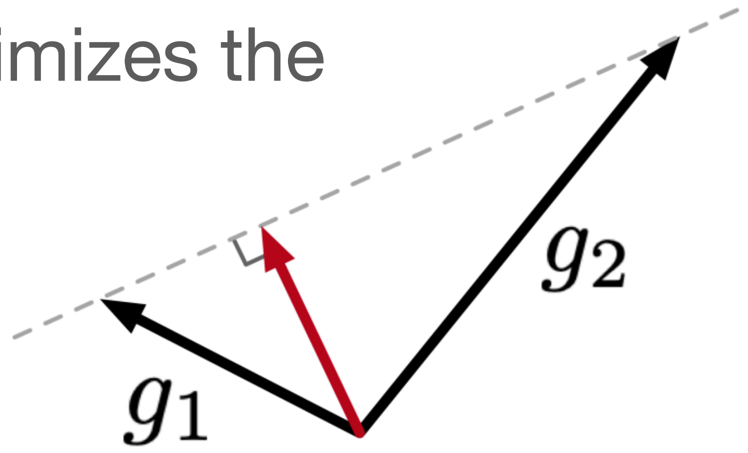
$$\max_{\|d\|_2 \le 1} \min_{i \in [k]} g_i^\top d$$

Interpretation: finding a common direction d that maximizes the worst-task improvement

Dual problem:

$$\min_{\alpha \in \Delta_{k-1}} \| \sum_{i=1}^{k} \alpha_i g_i \|_2^2$$

Interpretation: finding a convex combination α of the multiple gradients with minimum ℓ_2 norm



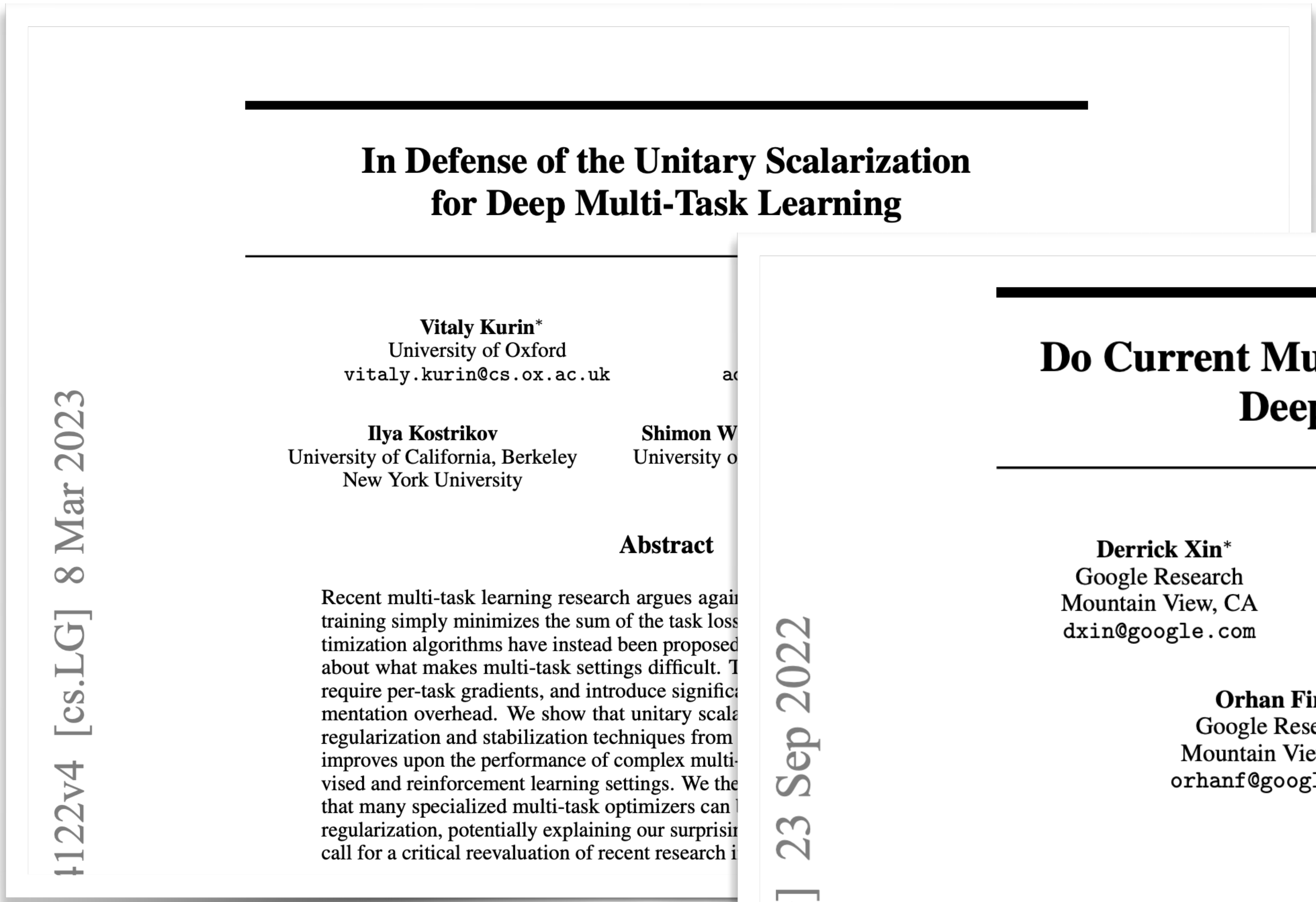
MGDA

$$\max_d \min_i g_i^\top d \quad \text{s.t.} \quad \|d\| \le 1$$

“Multiple-Gradient Descent Algorithm (MGDA) for Multi-Objective Optimization”, Comptes Rendus Mathématique, Désidéri, 2012.

Linear Scalarization vs Multi-Objective Optimization

Empirical comparisons between linear scalarization vs multi-objective optimization in NNs:



In Defense of the Unitary Scalarization for Deep Multi-Task Learning

Vitaly Kurin*
University of Oxford
vitaly.kurin@cs.ox.ac.uk

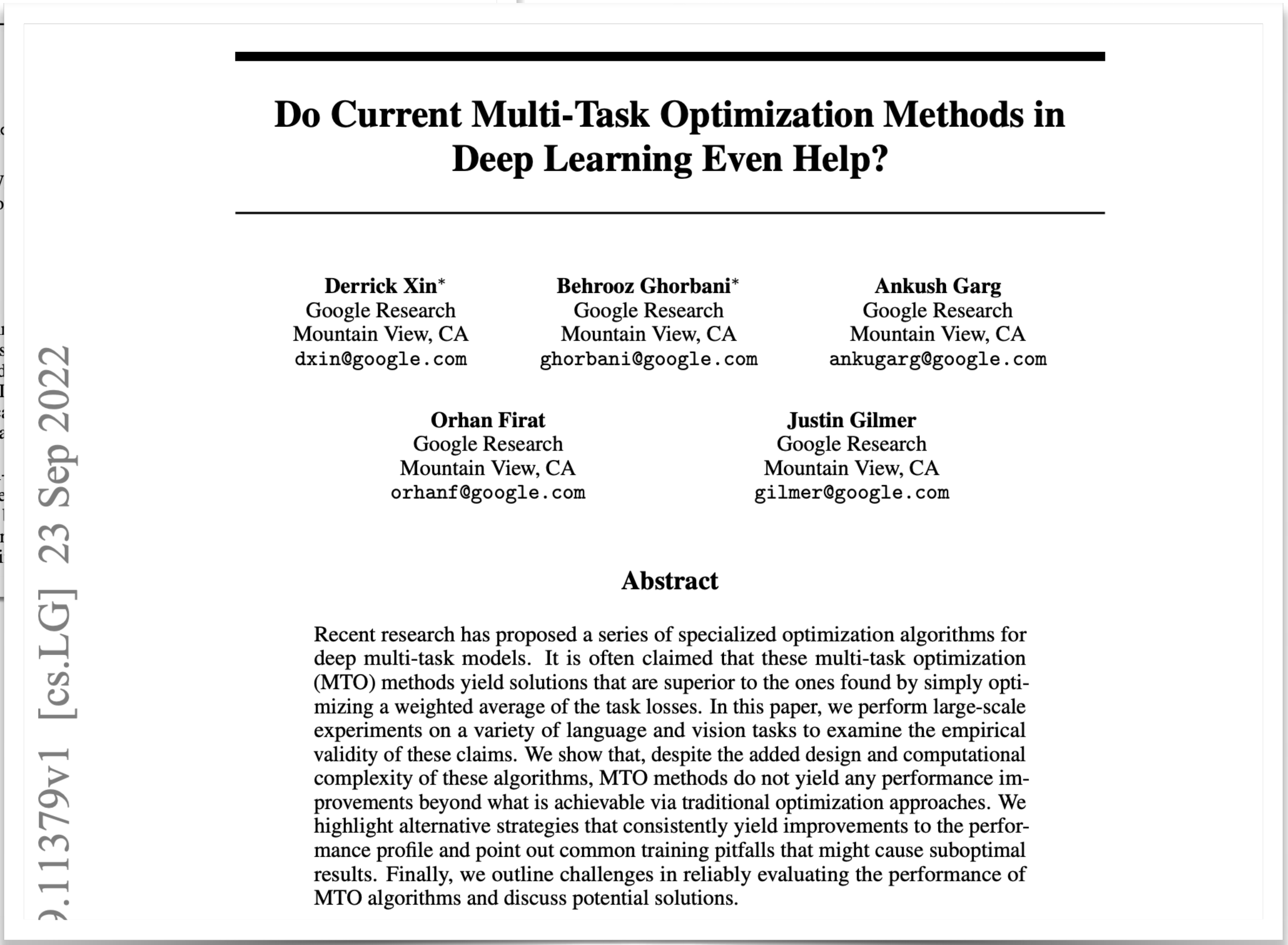
ac

Ilya Kostrikov
University of California, Berkeley
New York University

Shimon W
University o

Abstract

Recent multi-task learning research argues agai
training simply minimizes the sum of the task loss
timization algorithms have instead been proposed
about what makes multi-task settings difficult. T
require per-task gradients, and introduce significa
mentation overhead. We show that unitary scala
regularization and stabilization techniques from
improves upon the performance of complex multi-
vised and reinforcement learning settings. We the
that many specialized multi-task optimizers can l
regularization, potentially explaining our surprisi
call for a critical reevaluation of recent research i



Do Current Multi-Task Optimization Methods in Deep Learning Even Help?

Derrick Xin*
Google Research
Mountain View, CA
dxin@google.com

Behrooz Ghorbani*
Google Research
Mountain View, CA
ghorbani@google.com

Ankush Garg
Google Research
Mountain View, CA
ankugarg@google.com

Orhan Firat
Google Research
Mountain View, CA
orhanf@google.com

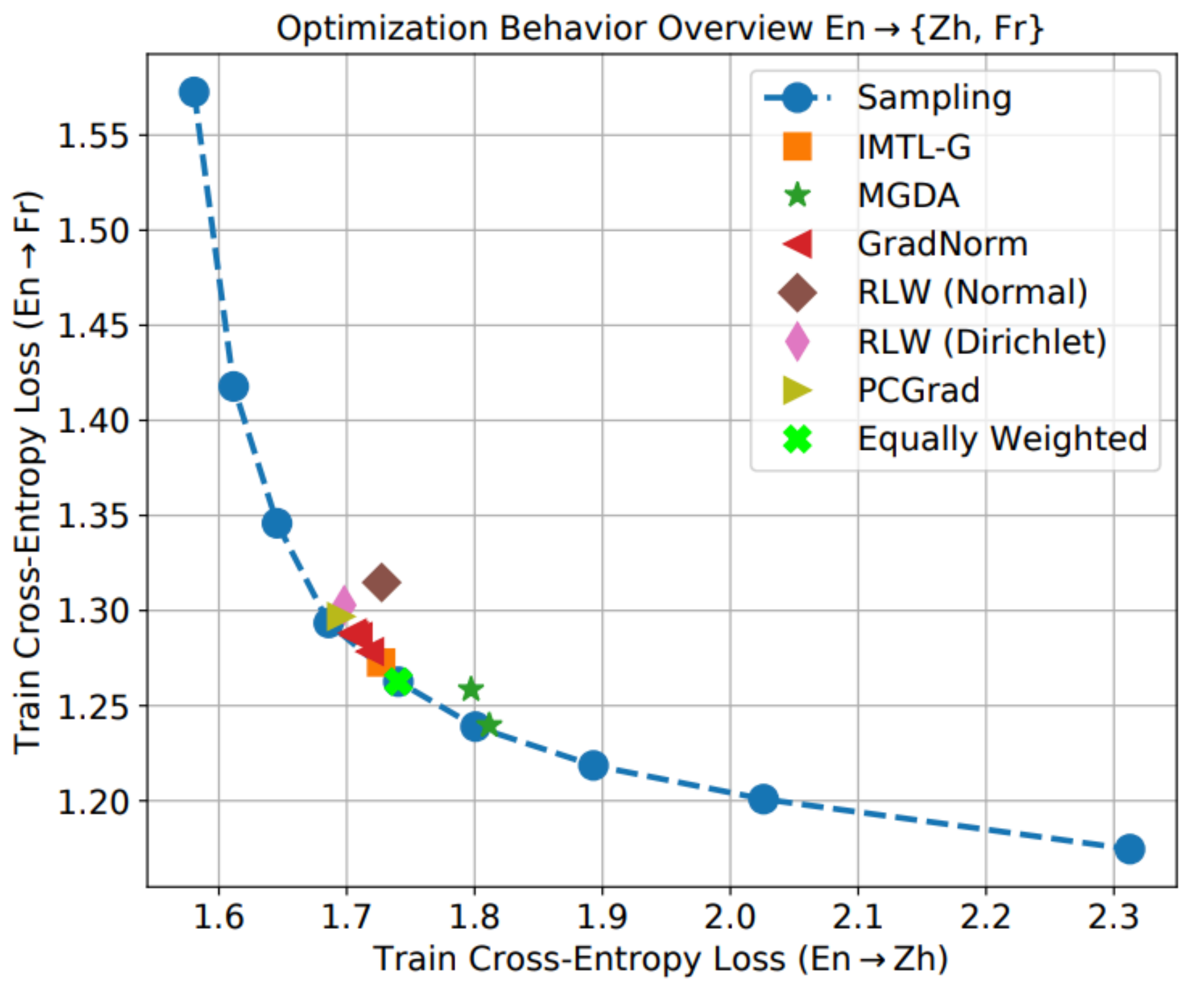
Justin Gilmer
Google Research
Mountain View, CA
gilmer@google.com

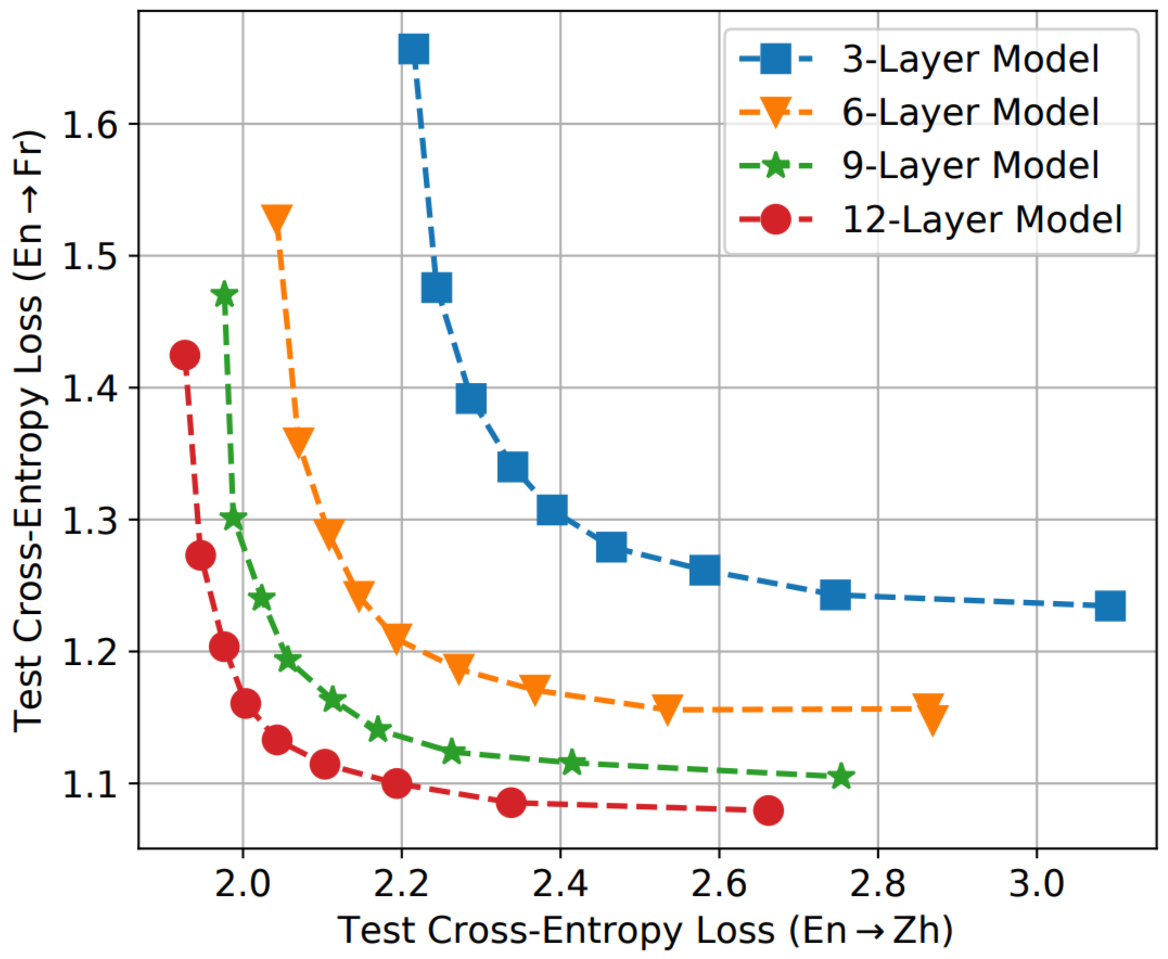
Abstract

Recent research has proposed a series of specialized optimization algorithms for deep multi-task models. It is often claimed that these multi-task optimization (MTO) methods yield solutions that are superior to the ones found by simply optimizing a weighted average of the task losses. In this paper, we perform large-scale experiments on a variety of language and vision tasks to examine the empirical validity of these claims. We show that, despite the added design and computational complexity of these algorithms, MTO methods do not yield any performance improvements beyond what is achievable via traditional optimization approaches. We highlight alternative strategies that consistently yield improvements to the performance profile and point out common training pitfalls that might cause suboptimal results. Finally, we outline challenges in reliably evaluating the performance of MTO algorithms and discuss potential solutions.

Linear Scalarization vs Multi-Objective Optimization

Empirical comparisons between linear scalarization vs multi-objective optimization in NNs:





Observations:

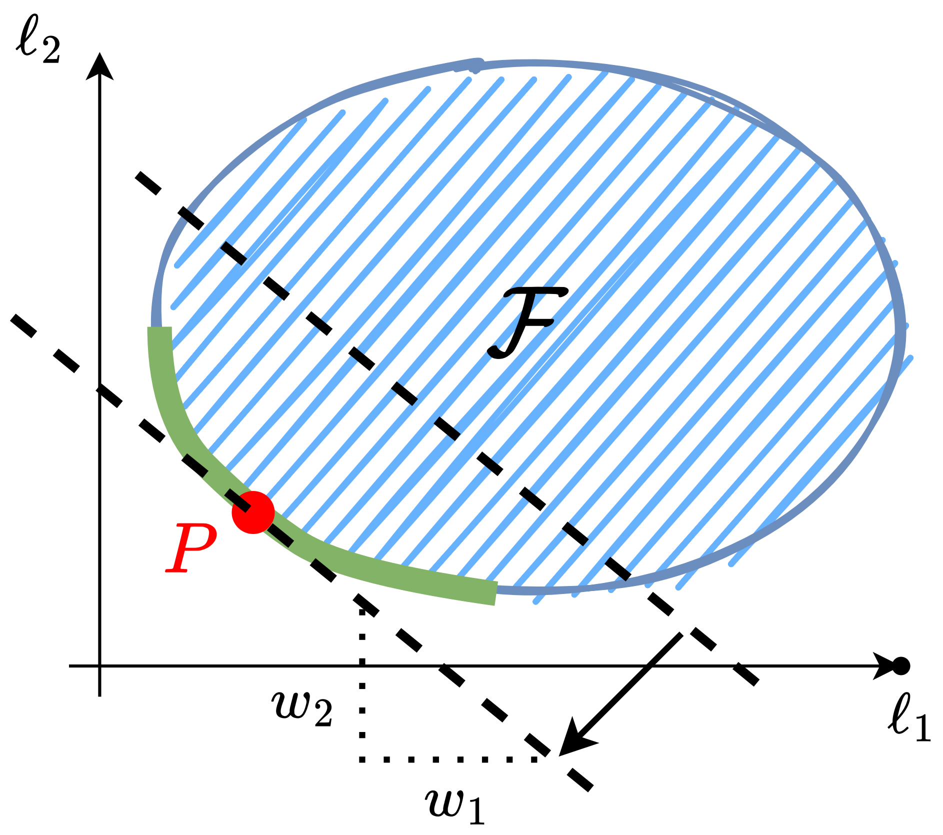
- For large-scale NNs, the closure of the Pareto front seems to be convex
- Deeper and larger models help to enlarge the feasible region (hence achieving Pareto-dominating solutions)

"Do Current Multi-Task Optimization Methods in Deep Learning Even Help?", NeurIPS' 22, Xin et al.

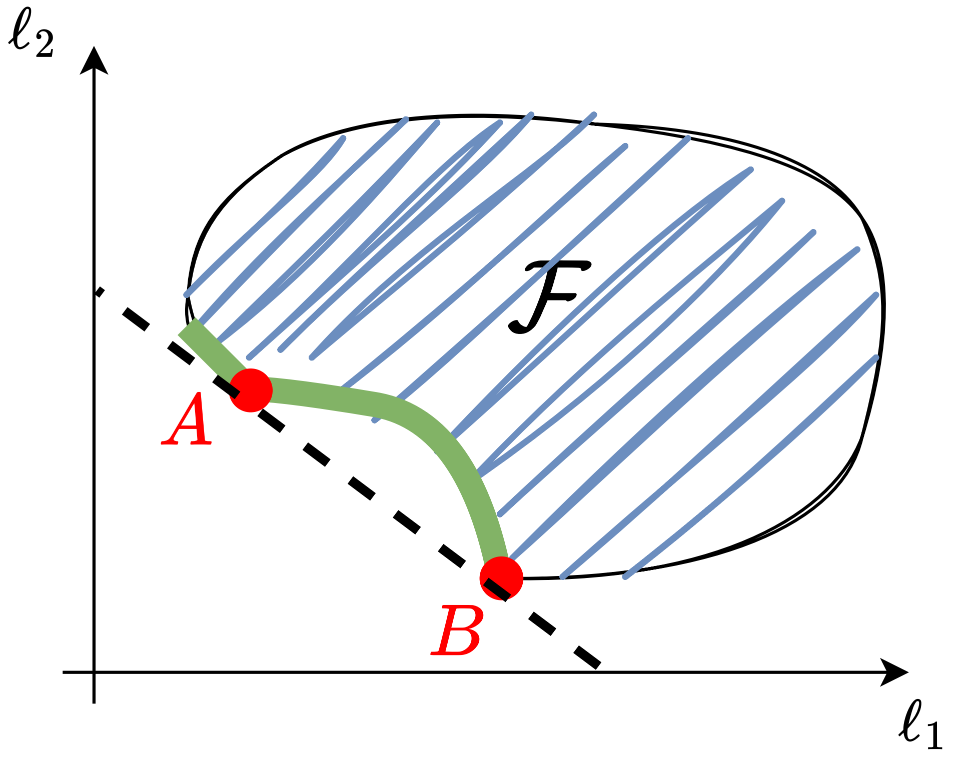
Linear Scalarization vs Multi-Objective Optimization

Research Question:

"For NNs, for every Pareto optimal $v \in \mathscr{P}(\mathscr{F})$, does there exists a $w \in \Delta_{k-1}$ such that the optimal solution of the linear scalarization problem corresponds to v?"



$\mathscr{P}(\mathscr{F}) = \mathscr{P}(\mathrm{conv}(\mathscr{F}))$



$\mathscr{P}(\mathscr{F}) \prec \mathscr{P}(\mathrm{conv}(\mathscr{F}))$

- When may MOO help?
- What is the impact of model size?

Linear Scalarization vs Multi-Objective Optimization

Multi-task linear network for regression:

- For each task $i \in [k]$, the prediction is given by

$$f_i(x, W, a_i) = x^\top W a_i$$

- Shared input $X \in \mathbb{R}^{n \times p}$, target vector $y_i \in \mathbb{R}^n, \forall i \in [k]$, the training loss for task i:

$$\ell_i(W, a_i) = \|XWa_i - y_i\|_2^2$$

- Parameter $W \in \mathbb{R}^{p \times q}$, i.e., network width = q, $a_i \in \mathbb{R}^q$ are task-specific parameters

Note:

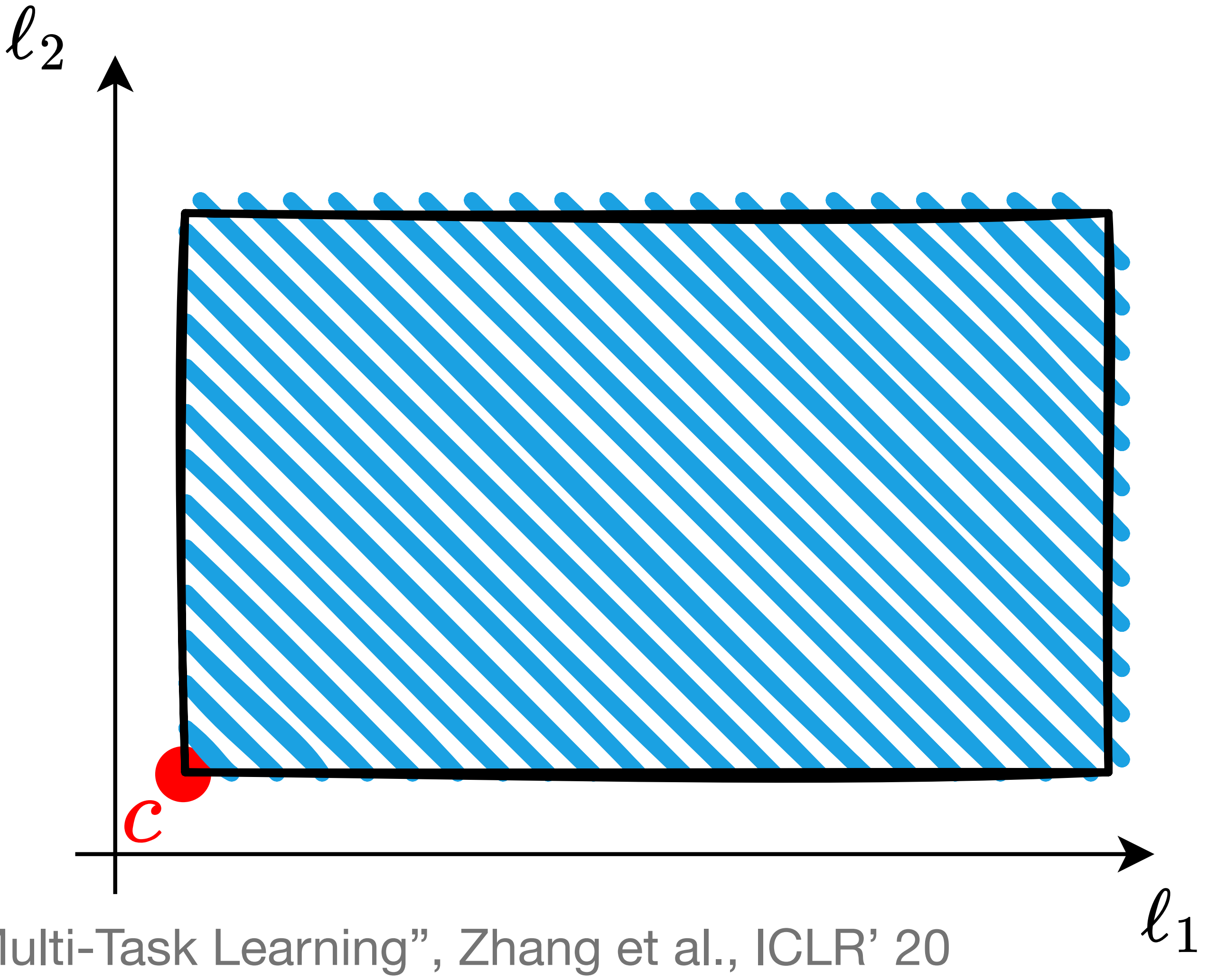
- For linear networks without loss of generality it suffices to consider a two layer network
- The overall model parameters $\theta = (\{a_i\}_{i=1}^k, W)$. The optimization problem is non-convex for each task i

Linear Scalarization vs Multi-Objective Optimization

Phase-transition between over-parametrized and under-parametrized networks:

Over-parametrized regime ($q \geq k$):

Theorem (informal): The network has sufficient capacity to fit all the tasks optimally, and the Pareto front reduces to a singleton $\mathscr{P}(\mathscr{F}) = \{c\}, c \in \mathbb{R}^k$ and hence can be attained via an arbitrary choice of convex coefficient $w \in \Delta_{k-1}$.



"Understanding and Improving Information Transfer in Multi-Task Learning", Zhang et al., ICLR' 20

"Revisiting Scalarization in Multi-Task Learning: A Theoretical Perspective", Hu et al., NeurIPS' 23

Linear Scalarization vs Multi-Objective Optimization

Phase-transition between over-parametrized and under-parametrized networks:

Over-parametrized regime ($q \geq k$):

Theorem (informal): The network has sufficient capacity to fit all the tasks optimally, and the Pareto front reduces to a singleton $\mathscr{P}(\mathscr{F}) = \{c\}, c \in \mathbb{R}^k$ and hence can be attained via an arbitrary choice of convex coefficient $w \in \Delta_{k-1}$.

Intuition: let $\hat{y}_i := X(X^\top X)^\dagger X^\top y_i$ be the optimal linear predictor for task i and let $\hat{Y} = [\hat{y}_1, \ldots, \hat{y}_k] \in \mathbb{R}^{n\times k}$ be a stack of column vectors. Then, due to the ℓ_2 loss, each task-specific loss can be decomposed as

$$\ell_i(W, a_i) = \|XWa_i - y\|_2^2 = \underbrace{\|XWa_i - \hat{y}_i\|_2^2}_{\text{Fitting error: can choose } a_i \text{ separately for a given shared } W} + \underbrace{\|\hat{y}_i - y_i\|_2^2}_{\text{Approximation error: projection loss (irreducible)}}$$

Fitting error: can choose a_i separately for a given shared W

Approximation error: projection loss (irreducible)

Linear Scalarization vs Multi-Objective Optimization

Over-parametrized regime ($q \geq k$):

Intuition: let $\hat{y}_i := X(X^\top X)^\dagger X^\top y_i$ be the optimal linear predictor for task i and let $\hat{Y} = [\hat{y}_1, \ldots, \hat{y}_k] \in \mathbb{R}^{n\times k}$ be a stack of column vectors. Then, due to the ℓ_2 loss, each task-specific loss can be decomposed as

$$\ell_i(W, a_i) = \|XWa_i - y\|_2^2 = \|XWa_i - \hat{y}_i\|_2^2 + \|\hat{y}_i - y_i\|_2^2$$

Note that $\hat{y}_i \in \mathrm{Col}(X)$ is the projection of y_i into the column space spanned by $X \in \mathbb{R}^{n\times p}$, i.e., the optimal linear prediction. Then if the network is wide enough, i.e., $q \geq k$, we can:

- Optimize the network parameter $W \in \mathbb{R}^{p\times q}$ and $\{a_i\}_{i=1}^k \subseteq \mathbb{R}^q$ by allocating one neuron for each task fitting loss $\hat{y}_i$
- Pick $W \in \mathbb{R}^{p\times q}$ such that W is full column-rank
- For every $i \in [k]$, $XWa_i = \hat{y}_i$ has a solution in terms of $a_i \in \mathbb{R}^q$ (because $q \geq k$ and $\hat{y}_i \in \mathrm{Col}(\hat{Y}) = \mathrm{Col}(XW)$)
- Putting all together, we have $\|XWa_i - \hat{y}_i\|_2^2 = 0$ and $c_i = \|\hat{y}_i - y_i\|_2^2$, $\forall i \in [k]$

Linear Scalarization vs Multi-Objective Optimization

Under-parametrized regime ($q < k$):

Theorem (informal): We focus on two extremal cases:

- Extremely under-parametrized ($q = 1$): Linear scalarization suffices, i.e., full-exploration of the Pareto front, if and only if $G := \hat{Y}^\top \hat{Y}$ is doubly non-negative, i.e., the inner products for all task pairs $\hat{y}_i$ and $\hat{y}_j$ are non-negative, up to negating the directions of some $\hat{y}_i, i \in [k]$
- Mildly under-parametrized ($q = k - 1$): Linear scalarization suffices if and only if $Q = G^{-1}$ is doubly non-negative, up to negating the directions of some $\hat{y}_i, i \in [k]$

Remark:

- This means that in general under the under-parametrized regime, linear scalarization is not sufficient of full exploration
- G and Q could be understood as a notion of "task-similarity" — task similarity is model-dependent!, i.e., $G_{ij} = \langle \hat{y}_i, \hat{y}_j \rangle$
- Sufficient and necessary conditions for the general case of $1 < q < k - 1$ still open

Linear Scalarization vs Multi-Objective Optimization

Geometric intuition of the under-parametrized regime:

Notation: let $\hat{y}_i := X(X^\top X)^\dagger X^\top y_i$ be the optimal linear predictor for task i and let $\hat{Y} = [\hat{y}_1, \ldots, \hat{y}_k] \in \mathbb{R}^{n \times k}$ be a stack of column vectors.

For every fixed $W \in \mathbb{R}^{p \times q}$, $Z = XW \in \mathbb{R}^{n \times q}$ are the linear representations learned by NNs. Each task-specific head admits an optimal solution $a_i^* = (Z^\top Z)^\dagger Z^\top \hat{y}_i$.

Hence, each task-specific loss ℓ_i could be simplified to

$$\min_{Z=XW} \|Z(Z^\top Z)^\dagger Z^\top \hat{y}_i - \hat{y}_i\|_2^2$$

Let $P_Z := Z(Z^\top Z)^\dagger Z^\top$ be the projection matrix under a fixed linear representation $Z = XW$, then the MOO optimization problem becomes

$$\max_{P_Z} \quad (\hat{y}_1^\top P_Z \hat{y}_1, \ldots, \hat{y}_k^\top P_Z \hat{y}_k)$$

Linear Scalarization vs Multi-Objective Optimization

Geometric intuition of the under-parametrized regime:

Let $P_Z := Z(Z^\top Z)^\dagger Z^\top$ be the projection matrix under a fixed linear representation $Z = XW$, then the MOO optimization problem becomes

$$\max_{P_Z} \quad (\hat{y}_1^\top P_Z \hat{y}_1, \ldots, \hat{y}_k^\top P_Z \hat{y}_k)$$

To illustrate the idea, let's consider the case $q = 1$ and $P_Z = vv^\top$ with $\|v\|_2 = 1$. Define $s_i = \hat{y}_i^\top v$ and $s = \hat{Y}^\top v \in \mathbb{R}^k$.

$$\Big\Downarrow$$

$$\max_{P_Z} \quad (\hat{y}_1^\top P_Z \hat{y}_1, \ldots, \hat{y}_k^\top P_Z \hat{y}_k) \iff \max_{v} \quad (s_1^2, \ldots, s_k^2)$$

But,

$$s^\top \left(\hat{Y}^\top \hat{Y}\right)^\dagger s = v^\top \hat{Y} \left(\hat{Y}^\top \hat{Y}\right)^\dagger \hat{Y}^\top v \le 1$$

This is a function of a k-dim ellipsoid!

Linear Scalarization vs Multi-Objective Optimization

Geometric intuition of the under-parametrized regime:

$$\max_{P_Z} \quad (\hat{y}_1^\top P_Z \hat{y}_1, \ldots, \hat{y}_k^\top P_Z \hat{y}_k) \iff \max_{v} \quad (s_1^2, \ldots, s_k^2)$$

and

$$s^\top \left(\hat{Y}^\top \hat{Y}\right)^\dagger s = v^\top \hat{Y} \left(\hat{Y}^\top \hat{Y}\right)^\dagger \hat{Y}^\top v \leq 1$$

This is a function of a k-dim ellipsoid!

Furthermore, a few observations:

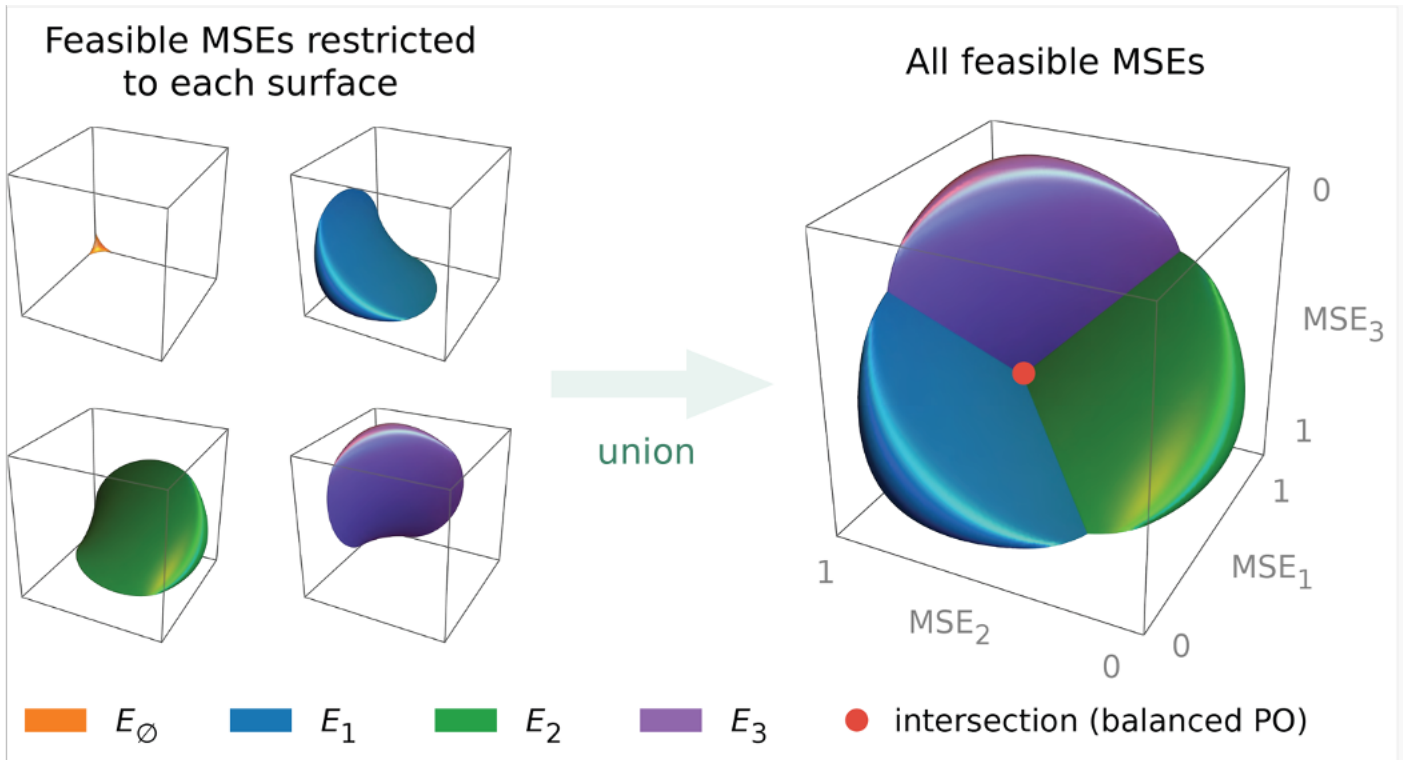
- The objective is invariant under negation of $\hat{y}_i, i \in [k]$
- Under negation, there are 2^k different configurations, each configuration corresponds to an (potentially degenerated) ellipsoid
- The feasible region $\mathscr{F}$ will be the union of 2^k ellipsoid

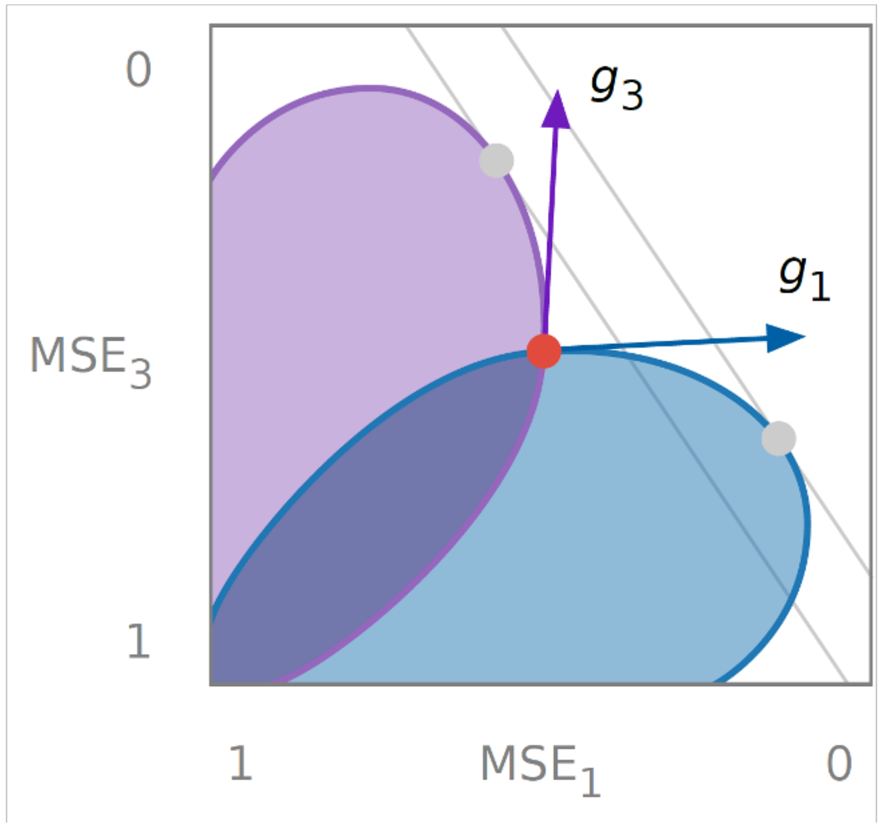
Linear Scalarization vs Multi-Objective Optimization

Geometric intuition of the under-parametrized regime:

Let $P_Z := Z(Z^\top Z)^\dagger Z^\top$ be the projection matrix under a fixed linear representation $Z = XW$, then the MOO optimization problem becomes

$$\max_{P_Z} \quad (\hat{y}_1 P_Z \hat{y}_1, \ldots, \hat{y}_k P_Z \hat{y}_k)$$





Linear Scalarization vs Multi-Objective Optimization

Multi-task non-linear network for regression:

Theorem (informal): If $q \geq nk$, then there exists a network that has sufficient capacity to fit all the tasks optimally, and the Pareto front reduces to a singleton $\mathscr{P}(\mathscr{F}) = \{c\}, c \in \mathbb{R}^k$ and hence can be attained via an arbitrary choice of convex coefficient $w \in \Delta_{k-1}$.

Remark:

- This upper bound is potentially very loose; ideally we would like an upper bound on the width q that only depends on the number of tasks k not the number of data points n
- No lower bound known, i.e., is there a trade-off problem in general for under-parametrized nonlinear NNs?

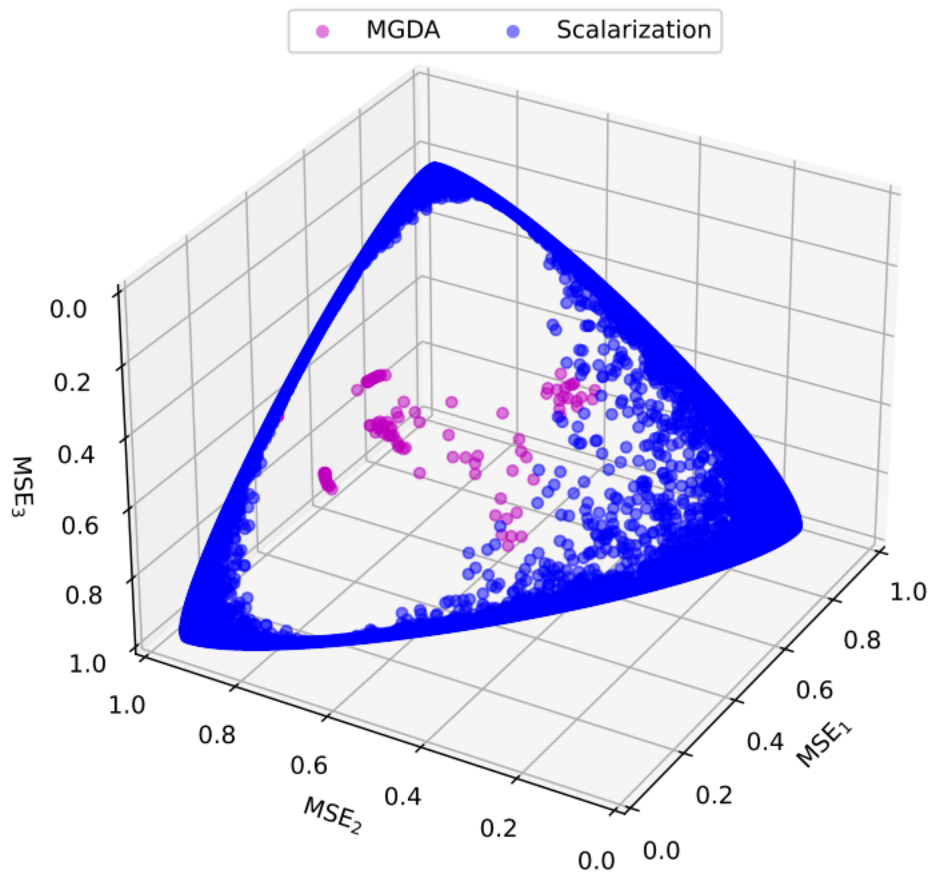
NN-Width	Linear NNs	Nonlinear NNs
Upper bound (sufficient)	k	nk
Lower bound (necessary)	k	?

Linear Scalarization vs Multi-Objective Optimization

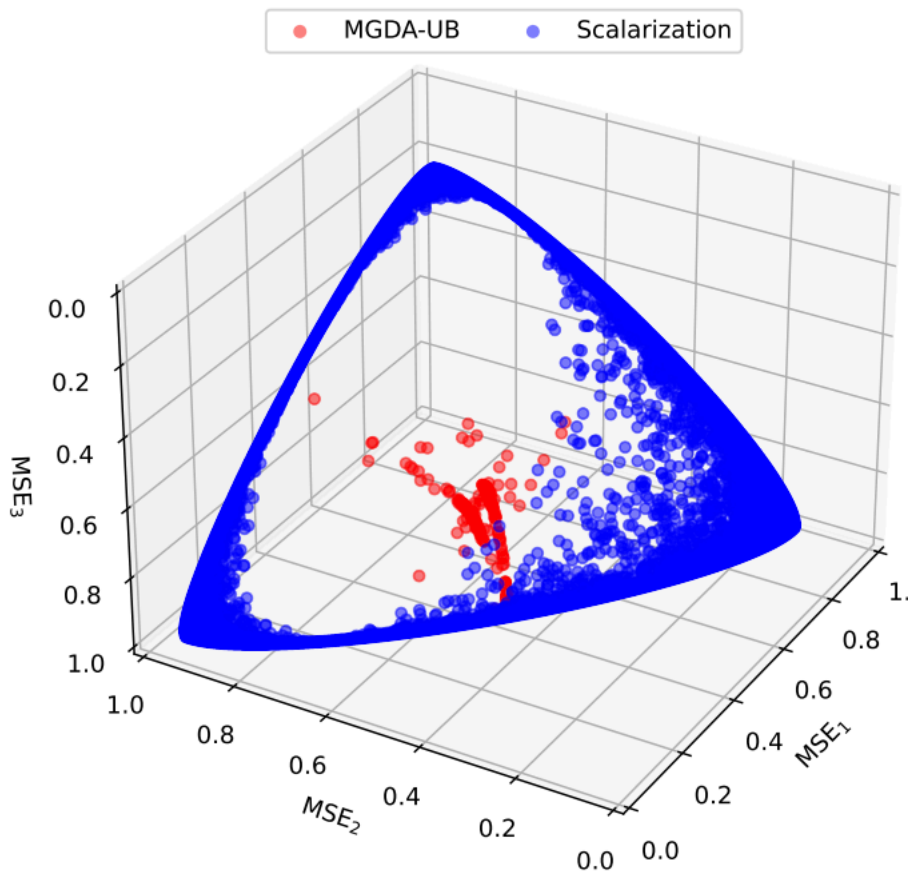
Multi-task non-linear network for regression:

Theorem (informal): If $q \geq nk$, then there exists a network that has sufficient capacity to fit all the tasks optimally, and the Pareto front reduces to a singleton $\mathscr{P}(\mathscr{F}) = \{c\}, c \in \mathbb{R}^k$ and hence can be attained via an arbitrary choice of convex coefficient $w \in \Delta_{k-1}$.

Empirical evidence:



(a) MGDA with multiple initializations



(b) MGDA-UB with multiple initializations

How to Rescue?

Under-parametrized regime ($q < k$): how to rescue?

Randomization!

Randomization ≈ Convexification

Given two under-parametrized networks f_0 and f_1, we can construct a randomized network as follows:

$$f(x) = \begin{cases} f_0(x) & \text{if } S \le t \\ f_1(x) & \text{o.w.} \end{cases}$$

where $t \in [0,1]$ and $S \sim U(0,1)$ is a uniform RV over $(0,1)$. Then

$$\mathbb{E}_{S,X,Y}[\ell(f(X), Y)] = t\mathbb{E}_{X,Y}[\ell(f_0(X), Y)] + (1-t)\mathbb{E}_{X,Y}[\ell(f_1(X), Y)]$$

By choosing different $t \in (0,1)$ we can interpolate and hence convexity any given feasible region $\mathscr{F}$.

$$\mathscr{F}_S = \mathrm{conv}(\mathscr{F}) \Longrightarrow \mathscr{P}(\mathscr{F}_S) = \mathscr{P}(\mathrm{conv}(\mathscr{F}))$$

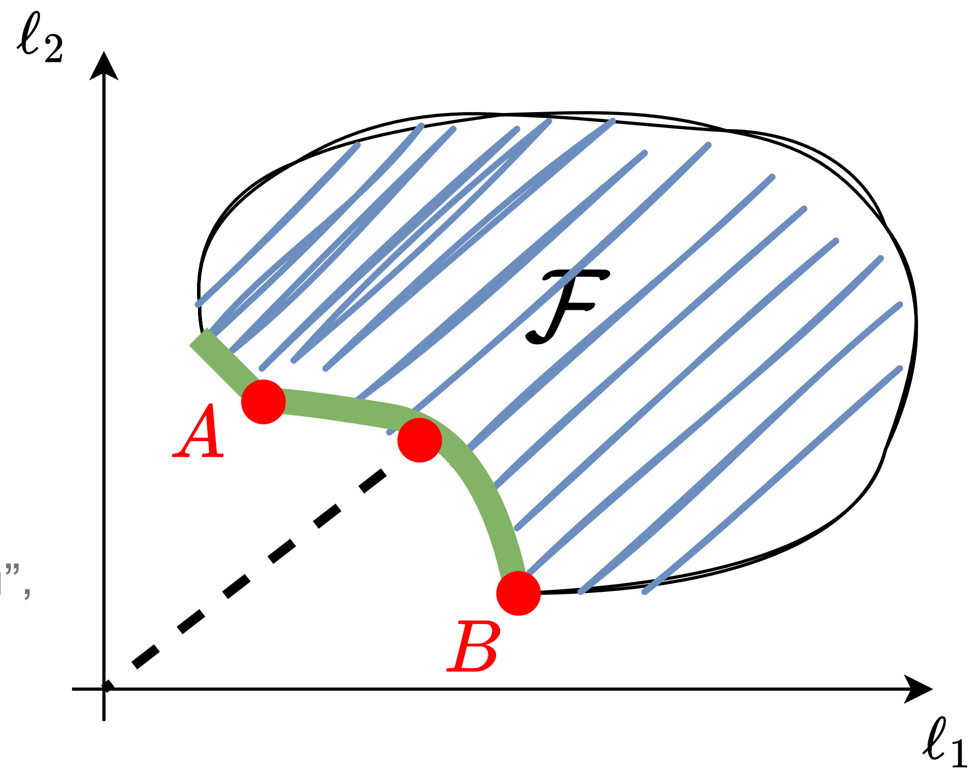
How to Rescue?

Under-parametrized regime ($q < k$): how to rescue?

Chebyshev scalarization: $w \in \Delta_{k-1} := \left\{ v \in \mathbb{R}^k : \sum_{i=1}^{k} v_i = 1, v_i \geq 0 \right\}$

$$\xrightarrow{\text{Chebyshev scalarization}} \quad \min_{g,h_i} \max_{i \in [k]} \frac{w_i}{n_i} \sum_{j=1}^{n_i} \ell_i((h_i \circ g)(x_j^{(i)}), y_j^{(i)})$$

Theorem (Choo & Atkins, 1983): Any feasible solution that is weakly Pareto optimal if and only if it is a solution for a weighted Chebyshev problem under some preference vector $w \in \Delta_{k-1}$.



"Proper Efficiency in Nonconvex Multicriterion Programming", Choo and Atkins, Mathematics of Operation Research, 1983
"A Unifying Perspective on Multi-Calibration: Game Dynamics for Multi-Objective Learning", Haghtalab et al., NeurIPS' 23
"Robust Multi-Task Learning with Excess Risks", He et al., ICML' 24
"Smooth Tchebycheff Scalarization for Multi-Objective Optimization", Lin et al., ICML' 24

Online Mirror Descent for Chebyshev Scalarization

Solving the Chebyshev scalarization problem with online mirror descent:

$$\min_{\theta\in\Theta}\max_{i\in[k]}\frac{1}{n_i}\sum_{j=1}^{n_i} w_i\ell_i(f_\theta(x_j^{(i)}), y_j^{(i)})$$

$$\Downarrow$$

$$\min_{\theta\in\Theta}\max_{\lambda\in\Delta_{k-1}}\frac{\lambda_i}{n_i}\sum_{j=1}^{n_i} w_i\ell_i(f_\theta(x_j^{(i)}), y_j^{(i)})$$

Upon receiving a batch $Z^{(t)} = \{X^{(t)}, Y^{(t)}\}$:

- Apply (projected) gradient descent to optimize the primal variable: model parameter $\theta \in \Theta$
- Apply exponentiated gradient / multiplicative weight update / hedging algorithm to optimize the dual variable: $\lambda \in \Delta_{k-1}$

"Online Mirror Descent for Tchebycheff Scalarization in Multi-Objective Optimization", Liu et al., arXiv: 2410.21764

Online Mirror Descent for Chebyshev Scalarization

Solving the Chebyshev scalarization problem with online mirror descent:

$$\min_{\theta \in \Theta} \max_{\lambda \in \Delta_{k-1}} \frac{\lambda_i}{n_i} \sum_{j=1}^{n_i} w_i \ell_i(f_\theta(x_j^{(i)}), y_j^{(i)})$$

Primal update:

$$\boldsymbol{\theta}^{(t+1)} = \Pi_\Theta \left(\boldsymbol{\theta}^{(t)} - \eta_{\boldsymbol{\theta}} \boldsymbol{\lambda}^{(t)} \circ \mathbf{w} \circ \nabla \mathbf{f}(\boldsymbol{\theta}^{(t)}) \right)$$

Dual update:

$$\lambda_i^{(t+1)} = \frac{\lambda_i^{(t)} \exp\left(\eta_{\boldsymbol{\lambda}} w_i f_i(\boldsymbol{\theta}^{(t)})\right)}{\sum_{j=1}^{m} \lambda_j^{(t)} \exp\left(\eta_{\boldsymbol{\lambda}} w_j f_j(\boldsymbol{\theta}^{(t)})\right)}$$

At the end of T iterations, we have a sequence of model parameters $\theta^{(1)}, \ldots, \theta^{(T)}$, how to combine them?

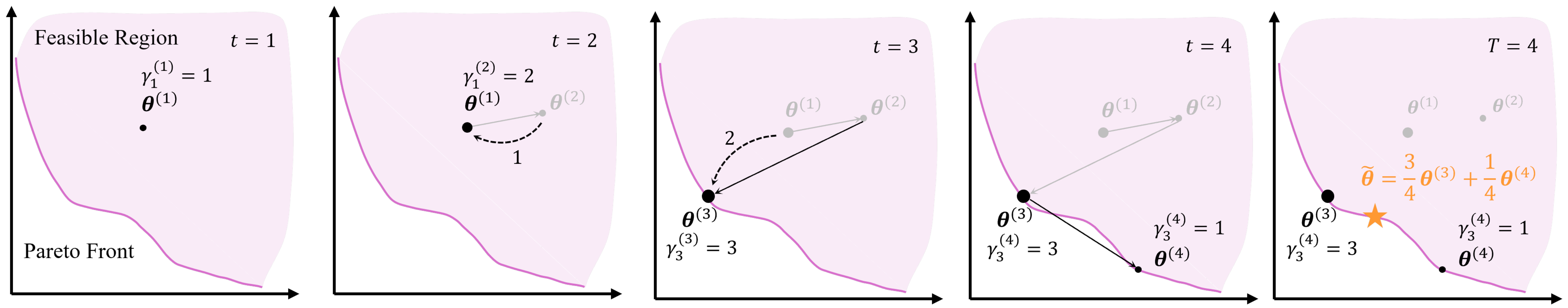
“Online Mirror Descent for Tchebycheff Scalarization in Multi-Objective Optimization”, Liu et al., arXiv: 2410.21764

Online Mirror Descent for Chebyshev Scalarization

Our solution: Adaptive online-to-batch conversion:

- Maintain an active set of PO solutions during the algorithm
- Credit assignment: weight $\gamma^{(t)}$ of each solution $\theta^{(t)}$ $\propto 1+$ # of intermediate solutions dominated by it
- For dominated solutions, weight $\gamma^{(t)} = 0$

$$\theta^* := \frac{1}{T}\sum_{t=1}^{T} \gamma^{(t)}\theta^{(t)}$$



Feasible Region
$t = 1$
$\gamma_1^{(1)} = 1$
$\boldsymbol{\theta}^{(1)}$
Pareto Front
$t = 2$
$\gamma_1^{(2)} = 2$
$\boldsymbol{\theta}^{(1)}$
$\boldsymbol{\theta}^{(2)}$
1
$t = 3$
$\boldsymbol{\theta}^{(1)}$
$\boldsymbol{\theta}^{(2)}$
2
$\boldsymbol{\theta}^{(3)}$
$\gamma_3^{(3)} = 3$
$t = 4$
$\boldsymbol{\theta}^{(1)}$
$\boldsymbol{\theta}^{(2)}$
$\boldsymbol{\theta}^{(3)}$
$\gamma_3^{(4)} = 3$
$\gamma_3^{(4)} = 1$
$\boldsymbol{\theta}^{(4)}$
$T = 4$
$\boldsymbol{\theta}^{(1)}$
$\boldsymbol{\theta}^{(2)}$
$\tilde{\boldsymbol{\theta}} = \frac{3}{4}\boldsymbol{\theta}^{(3)} + \frac{1}{4}\boldsymbol{\theta}^{(4)}$
$\boldsymbol{\theta}^{(3)}$
$\gamma_3^{(4)} = 3$
$\gamma_3^{(4)} = 1$
$\boldsymbol{\theta}^{(4)}$

Online Mirror Descent for Chebyshev Scalarization

Under the following assumptions:

- Convexity: each $f_i(\theta)$ is convex in $\theta \in \mathbb{R}^d$
- Bounded feasible region: $\forall \theta \in \Theta, \|\theta\|_2 \leq R_\theta$
- Bounded gradients: $\forall i \in [k], \forall \theta \in \Theta, \|\nabla_\theta f_i(\theta)\|_2 \leq L$

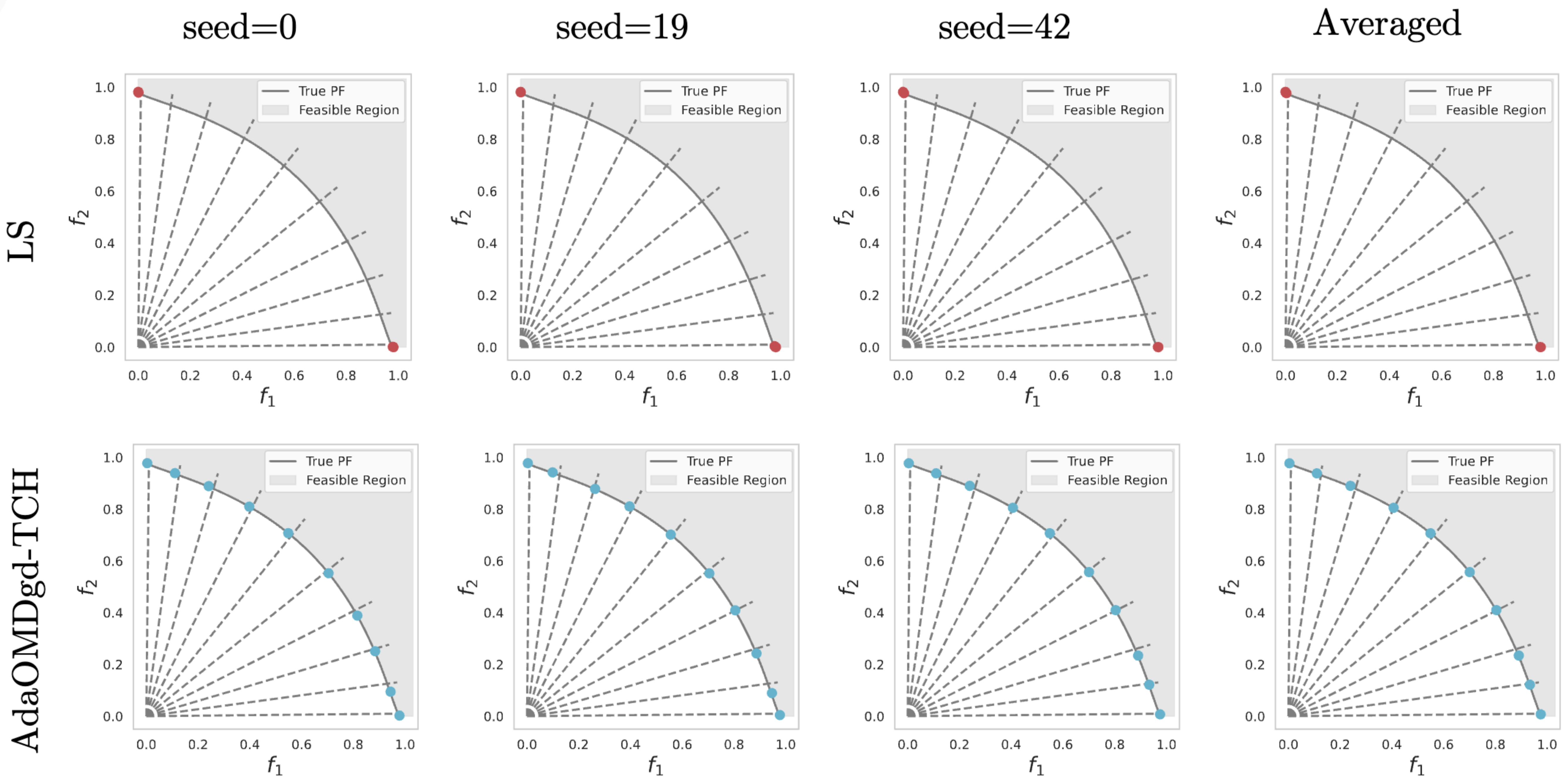
$$\mathrm{TCH}(\theta; w) := \max_{i \in [k]} \frac{1}{n_i} \sum_{j=1}^{n_i} w_i \ell_i(f_\theta(x_j^{(i)}), y_j^{(i)})$$

Theorem (convergence): under the above assumptions and the adaptive online mirror descent with learning rate $\eta = O(\sqrt{1/T})$, the algorithm converges as follows:

$$\mathbb{E}\left[\mathrm{TCH}(\theta^*; w)\right] - \min_{\theta \in \Theta} \mathrm{TCH}(\theta; w) \leq O\left(\frac{d}{\sqrt{T}} + \frac{\sqrt{\log k}}{\sqrt{T}}\right)$$

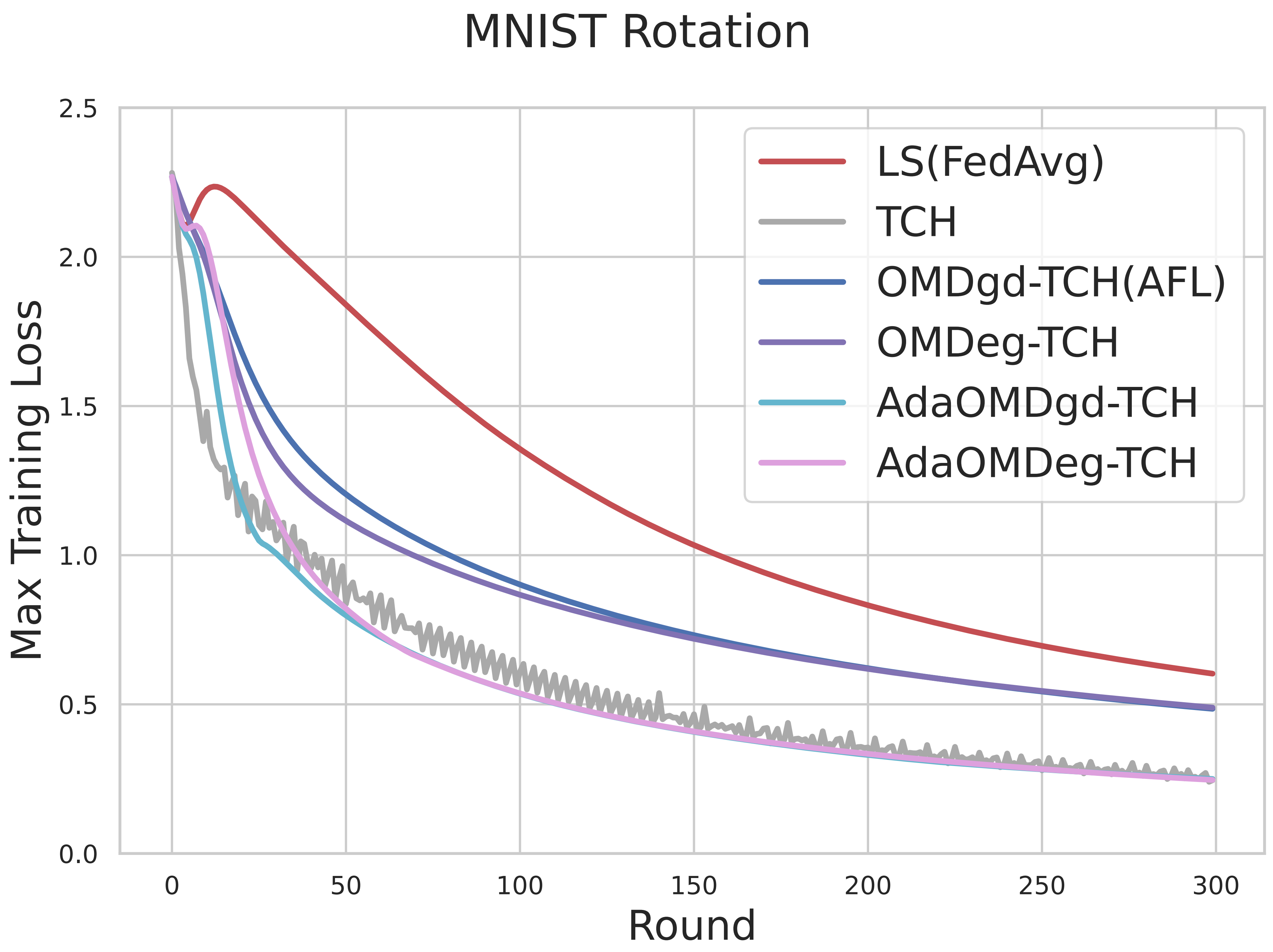
Online Mirror Descent for Chebyshev Scalarization

Controlled PO solution:



Online Mirror Descent for Chebyshev Scalarization

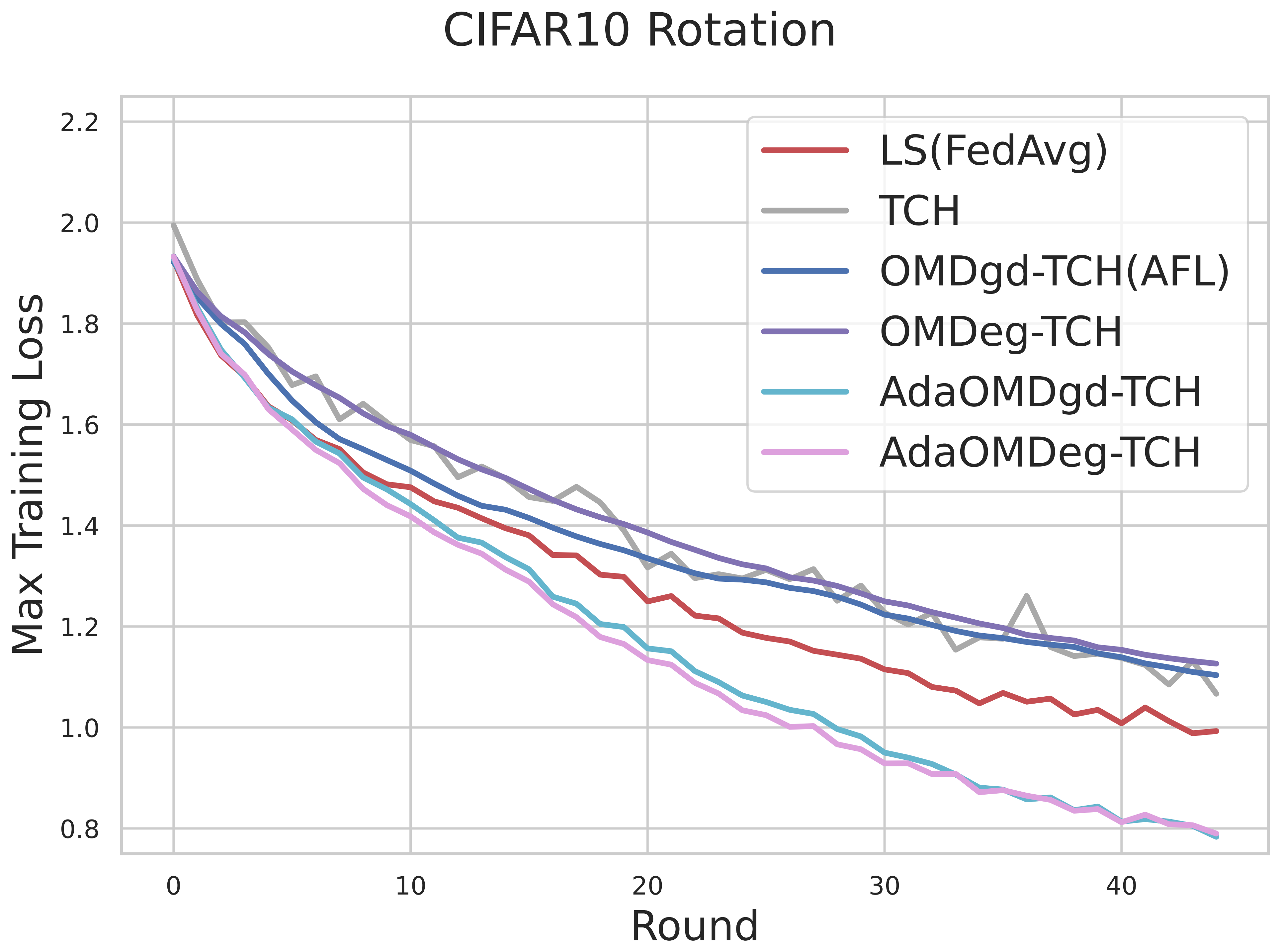
Convergence speed and stability:



“Online Mirror Descent for Tchebycheff Scalarization in Multi-Objective Optimization”, Liu et al., arXiv: 2410.21764

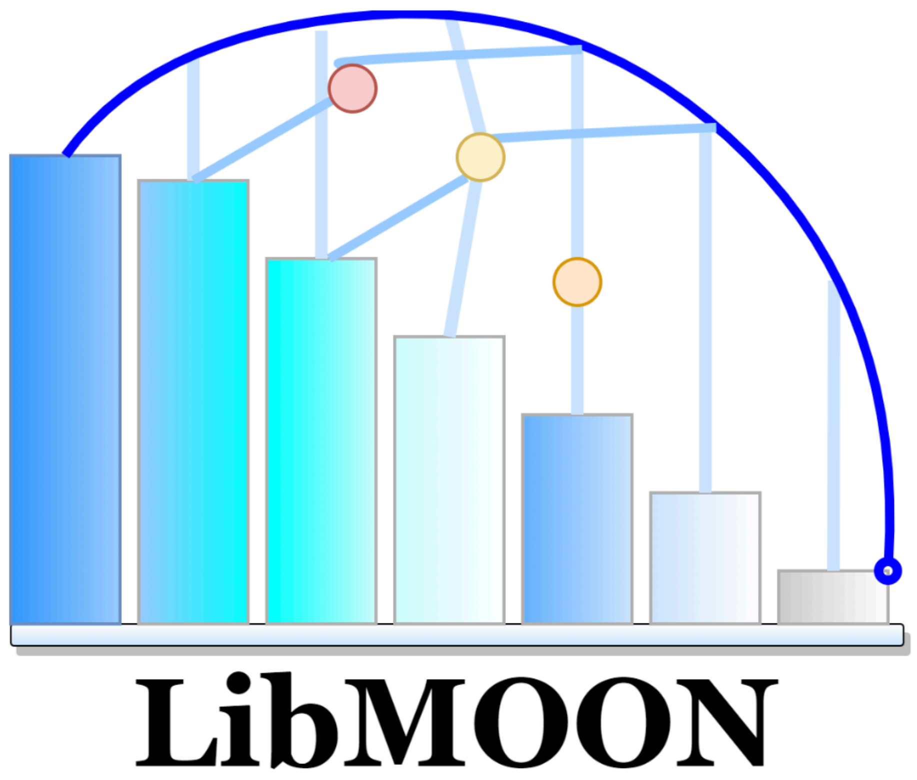
Online Mirror Descent for Chebyshev Scalarization

Convergence speed and stability:



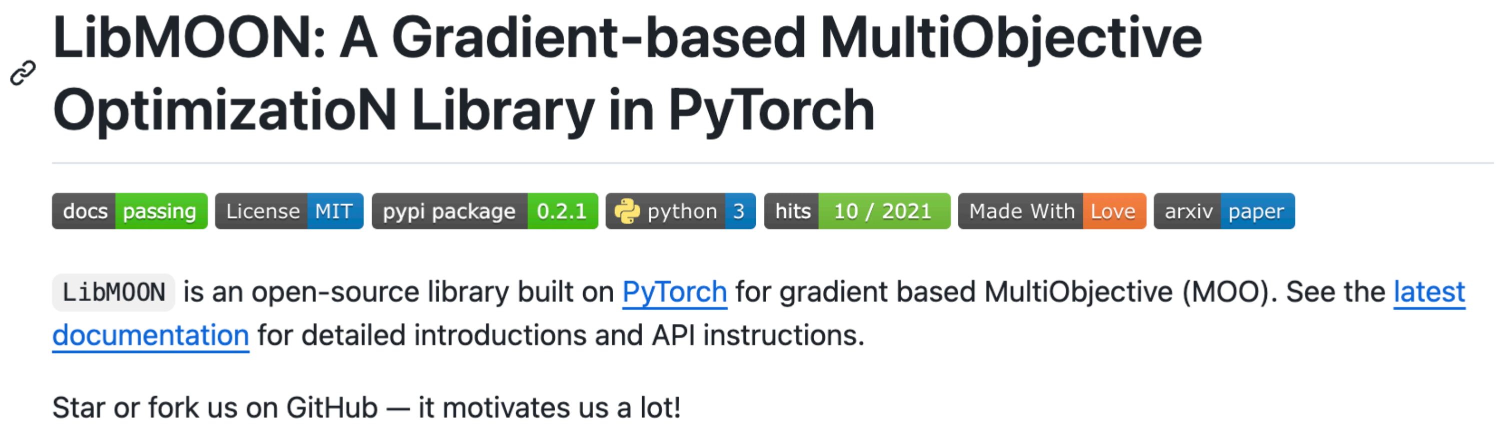
“Online Mirror Descent for Tchebycheff Scalarization in Multi-Objective Optimization”, Liu et al., arXiv: 2410.21764

LibMoon: A Gradient-based MultiObjective OptimizatioN Library in PyTorch



Method	Solution Property	Complexity	Pref.
EPO [16]	Exact solutions	$O(m^2nK)$	✓
HVGrad [37]	Solutions with maximal HV	$O(m^2nK^2)$	×
MGDA-UB [12]	Random solutions	$O(m^2nK)$	×
MOO-SVGD [17]	Diversity by particles repulsion	$O(m^2nK^2)$	×
PMGDA [38]	Solutions under specific demands	$O(m^2nK)$	✓
PMTL [13]	Solutions in sectors	$O(m^2nK^2)$	×
Random [39]	Random solutions	$O(m^2nK)$	×
Agg-LS [36]	Convex part of a PF	$O(mnK)$	✓
Agg-Tche [29]	Exact solutions	$O(mnK)$	✓
Agg-mTche [40]	Exact solutions	$O(mnK)$	✓
Agg-PBI [29]	Approximate exact solutions	$O(mnK)$	✓
Agg-COSMOS [33]	Approximate exact solutions	$O(mnK)$	✓
Agg-SmoothTche [22]	Approximate exact solutions	$O(mnK)$	✓

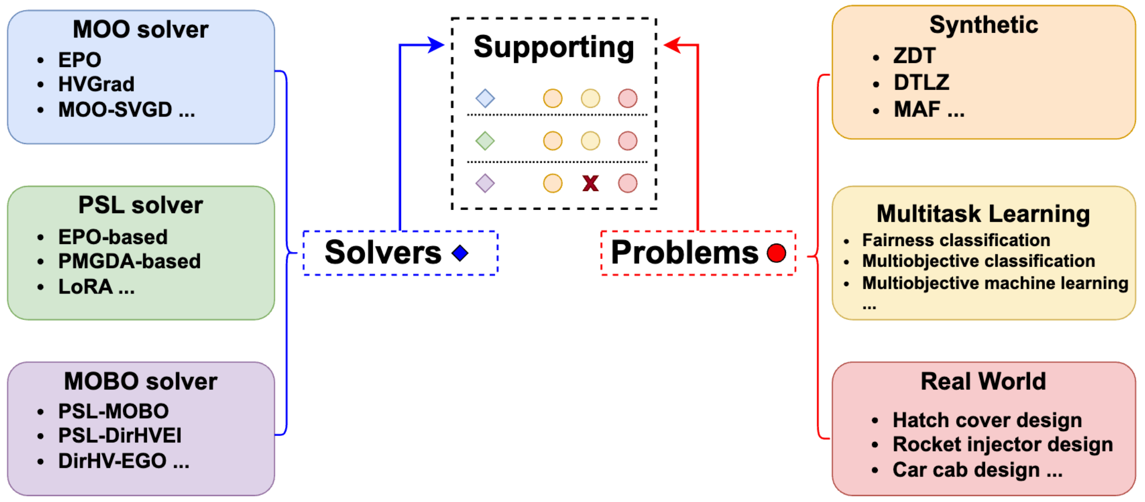
m: number of objectives. n: number of decision variables. K: number of subproblems. m is usually small (e.g., 2-4), K is relatively large (e.g., 20-40), and n is particularly large (e.g., 10,000). Therefore, m^2 is not a big concern, while K^2 and n^2 are big concerns. Complexity is for time complexity, and Pref. denotes whether this method is preference-based or not.


LibMOON: A Gradient-based MultiObjective OptimizatioN Library in PyTorch

docs passing | License MIT | pypi package 0.2.1 | python 3 | hits 10 / 2021 | Made With Love | arxiv paper

`LibMOON` is an open-source library built on PyTorch for gradient based MultiObjective (MOO). See the latest documentation for detailed introductions and API instructions.

Star or fork us on GitHub — it motivates us a lot!



"LibMOON: A Gradient-based MultiObjective OptimizatioN Library in PyTorch", Zhang et al., NeurIPS' 24 D&B Track
Github repo: https://github.com/xzhang2523/libmoon

Summary

Insights and Implications for us:

- The problem of linear scalarization vs MOO for multitask learning is model-dependent
- Good news: with sufficient capacity of the networks, linear scalarization can represent every Pareto optimal solution
- For linear MTL models under regression tasks we identify a precise phase-transition $q = k$
- For nonlinear MTL models we have a (loose) upper bound $q = nk$
- For under-parametrized models, we can use Chebyshev scalarization to control the converged PO solution

Open questions:

- How about linear MTL for classification?
- Is there a similar phase transition phenomenon for nonlinear MTL models?
- Pareto-set learning: a single model/algorithm to learn all the diverse PO solutions simultaneously

Thanks!

IBM Research

amazon

Meta

Google Research

DARPA

NSF National Science Foundation

Reference:

1. Revisiting Scalarization in Multi-Task Learning: A Theoretical Perspective, NeurIPS' 23
2. Robust Multi-Task Learning with Excess Risks, ICML' 24
3. LibMOON: A Gradient-based MultiObjective OptimizatioN Library in PyTorch, NeurIPS' 24 D&B Track
4. Online Mirror Descent for Tchebycheff Scalarization in Multi-Objective Optimization, arXiv: 2410.21764