Learning with Multiple Objectives - Foundations and Applications

Tianyi Chen    Zhuoran Yang    Lisha Chen

Rensselaer Polytechnic Institute
Yale University

Room 111, Vancouver Convention Centre – West Building
Vancouver, BC, Canada
2:00 pm - 6:00 pm, February 20, 2024
Tianyi Chen  
Rensselaer Polytechnic Institute  
Bilevel optimization, multi-objective optimization

Zhuoran Yang  
Yale University  
Reinforcement learning, deep learning, and statistics

Lisha Chen  
Rensselaer Polytechnic Institute  
Multi-objective optimization, and statistical learning
Outline

- Part I - Introduction and background (20 mins)
- Part II - Bilevel optimization fundamentals
- Part III - Bilevel applications to reinforcement learning
- Part IV - Multi-objective learning beyond bilevel optimization
- Part V - Conclusions and open directions
Success of AI before 2020

AI benchmark saturation over time

Time to saturation:
- MNIST: 15 years
- ImageNet: 6 years
- SQuAD 1.1: 2 years
- SQuAD 2.0: 1 year
- GLUE: 1 year
Success of AI before 2020
Success of AI before 2020

Excel at (only) one thing!
OpenAI
GPT-IQ
NEW ERA UNFOLDS!
Multiple tasks and data modalities arise today

To merge two dictionaries in Python, you can use the `update()` method.
Multiple metrics arise in machine learning today

Data and model bias

Resource constraints

Desiderata of multi-objective AI

Fast adaptation to new users

Subject to privacy regulation
"Past" era of single-objective learning

Empirical risk minimization (ERM)
\[
\min_x \text{loss} / \text{error} (\text{model } x, \text{training data})
\]

Data collection

Model deployment

Conventional machine learning (ML) pipeline
Tasks, data, metrics all can be modeled as an objective...

\[
\min_x \text{loss (model } x, \text{ training data, metric, tasks)}
\]
Tackling multiple tasks, data, metrics via single-objective learning …

\[
\begin{aligned}
\min_x & \text{ loss (model } x, \text{ )} \\
& + \\
\min_x & \text{ loss (model } x, \text{ )} \\
& + \\
\min_x & \text{ loss (model } x, \text{ )}
\end{aligned}
\]

- e.g., increase \( x^{(1)} \), increase acc 0.01
- e.g., increase \( x^{(1)} \), decrease fair 20%
- e.g., decrease \( x^{(1)} \), increase acc 0.1

Simple but may cause… unit mismatch or competition
Tackling multiple tasks, data, metrics via sequential learning …

\[
\min_x \text{ loss (model } x, \quad \text{ )}
\]

\[
\min_x \text{ loss (model } x, \quad \text{ )}
\]

\[
\min_x \text{ loss (model } x, \quad \text{ )}
\]

No feedback, may cause catastrophic forgetting
Our focus – A tale of two methods

Bi-/multi-level training

Pre-define the preferences/orders

\[
\begin{align*}
& \text{min } \text{objective 1} \\
& \text{s.t. } \text{min } \text{objective 2} \\
& \text{s.t. } \text{min } \text{objective 3}
\end{align*}
\]

Allow feedback loops, compared with sequential learning

Multi-objective training

Pre-define or let the algorithm determine preferences

\[
\text{min } (\text{objective 1, objective 2, objective 3})
\]

Mitigate unit mismatch or competition, compared with single-objective learning
Opportunity lies in new model training steps

Data collection

Model deployment

Multi-objective training

\[
\min_x \left[ \text{obj } 1(x), \text{obj } 2(x), \ldots \text{obj } M(x) \right]
\]

Hypothesis \( \mathcal{H} \)

Pareto front

\( \ell_1 \)

\( \ell_2 \)

New model training
Bi-level training

\[
\min_x \text{ second objective } (x, y^*(x)) \\
\begin{align*}
y^*(x) &= \arg\min_y \text{ first objective } (x, y) 
\end{align*}
\]

Problems tackled by bilevel model training?

Learning from imbalanced data
Learning to fast adapt
Learning to fast optimize
Neural architecture search
Adversarial training
Model pruning

...
Bilevel optimization for learning from non-i.i.d. data

Bilevel optimization for meta learning

\[ \min_{x} \text{ task validation data } (y^*(x)) \quad \text{s. t. } y^*(x) = \arg\min_{y} \text{ train data } (x, y) \]

**Upper level** optimizes the meta model \( x \)

**Lower-level** for \( x \)-fixed model fine-tuning \( y \)

What is bilevel optimization? A gentle introduction

Bilevel optimization can be defined as

\[
\begin{align*}
\min_{x \in \mathcal{X}, y} & \quad f(x, y) \quad \text{(upper level)} \\
\text{s.t.} & \quad y \in \arg\min_{y' \in \mathcal{Y}} g(x, y') \quad \text{(lower level)}
\end{align*}
\]

- **Merits**: capture learning hierarchy across multiple objectives
- **Difficulty**: upper- and lower-level coupling through solution set
Relation with other popular frameworks

\[
\min_{x \in \mathcal{X}, y} f(x, y)
\]

s.t. \( y \in \arg \min_{y' \in \mathcal{Y}} g(x, y') \)

▪ Bilevel versus min-max optimization

\[
g(x, y) := -f(x, y) \quad \text{min max} \quad f(x, y)
\]

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," ICLR 2018
Despite its flexibility, is it too slow to solve?

1. Introduction

A sequential optimization problem in which independent decision makers act in a noncooperative manner to maximize their individual benefits may be categorized as a Stackelberg game. The bilevel programming problem is a static, open-loop version of this game where the leader controls the decisions in a higher-level problem called the bilevel programming problem (BLPP). We begin with a pair of examples showing that, even under the best of circumstances, solutions may not exist. This is followed by a proof that the BLPP is NP-hard.

Key Words. Bilevel programming, Stackelberg games, computational
A brief history of bilevel optimization

1952 Stackelberg's game
1973 Original bilevel formulation
1980s Single-level reformulation of bilevel optimization
Early 1990s Hardness results
Late 1990s
Late 2000s
After 2020: Finite-time convergence; generalization; new AI/ML applications

Classification model selection via bilevel programming
G. KUNAPULI*, K. P. BENNETT†, JING HU* and JONG-SHI PANG‡

†Department of Mathematical Sciences, Rensselaer Polytechnic Institute, 110 8th Street, Troy NY 12180.
‡ Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, 104 S. Mathews Ave., Urbana IL 61801.

(Mathematical Programs with Equilibrium Constraints)

Mathematical Programs with Optimization Problems in the Constraints

Recent surge of interests

Papers on Google Scholar under keyword “bilevel optimization”
Problems tackled by multi-objective training?

Learning from multiple tasks
Multilingual translation
Multi-objective alignment
Multi-domain classification
Multi-agent reinforcement learning

Multi-objective training

\[ \min_{x} \left[ \text{obj}_1(x), \text{obj}_2(x), \ldots, \text{obj}_M(x) \right] \]
Multi-objective optimization for multilingual translation

Universal language translator over 7000 languages

\[
\min_x \left[ \text{Laguage } 1(x); \text{Laguage } 2(x); \ldots; \text{Laguage } 7000(x) \right]
\]

Y. Cheng, Y. Zhang, M. Johnson, W. Macherey, and A. Bapna, “Mu2slam: Multitask, multilingual speech and language models.” In International Conference on Machine Learning, pp. 5504–5520, 2023
Multi-objective optimization for multi-task robotics

What is multi-objective optimization?

“\[ \min_{x,y} F(x, y) = [f_1(x, y_1), \ldots, f_t(x, y_t), \ldots, f_T(x, y_T)] \]”

A vector optimization problem

- **Merits**: potentially capture all preferences/tradeoffs among objectives
- **Difficulty**: How to optimize or even compare a vector? *(see part 3)*

\[
\begin{bmatrix} 1 \\ 0 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix} ? \quad \begin{bmatrix} 2 \\ 0 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix} ?
\]
Relation with other popular frameworks

“\[ \min_{x,y} F(x, y) = [f_1(x, y), \ldots, f_2(x, y)] \]"

More general and flexible models!

- **Multi-objective** versus **functional constrained** optimization

\[
F_1 := \min_{x, y} f_1(x, y) \quad \text{s.t.} \quad f_2(x, y) \leq F_2
\]

change \( F_2 \) to obtain all \((F_1, F_2)\)
History of multi-objective optimization

1880s – 1900s
- Lexicographic
- Static weighted sum

1970s – present
- Bilevel and Multilevel Programming
- A Bibliography Review

1980s – present
- MGDA

2010s – present
- Application to AI/ML
- Uncertainty-based

MGDA
Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics

Application to AI/ML

Uncertainty-based

Errors
Training
Generalization
Model complexity
Recent surge of interests

Google Scholar under keyword “multi-objective optimization”
Rationale for this tutorial

- Part II: Recent advances in bilevel optimization foundations
- Part III: A representative application to reinforcement learning
- Part IV: Recent advances in multi-objective learning foundations
Tutorial Part II: Bilevel Optimization Fundamentals

Tianyi Chen

Rensselaer Polytechnic Institute

February 20, 2024
Part I - Introduction and background

Part II - Bilevel optimization fundamentals (60 mins)

Part III - Bilevel applications to reinforcement learning

Part IV - Multi-objective learning beyond bilevel optimization

Part V - Conclusions and open directions
Two general recipes for bilevel optimization

\[
\min_{x \in \mathcal{X}, y} \quad f(x, y)
\]
\[
\text{s.t.} \quad y \in S(x) := \arg \min_{y' \in \mathcal{Y}} \quad g(x, y')
\]

- **Nested optimization** first over \( y \) and then over \( x \)
  
  \[
  \min_{x \in \mathcal{X}} \quad F(x)
  \]
  \[
  \text{with} \quad F(x) := \min_{y \in S(x)} \quad f(x, y)
  \]

- **Constrained optimization** jointly over \( x \) and \( y \)
  
  \[
  \min_{x \in \mathcal{X}, y \in \mathcal{Y}} \quad f(x, y)
  \]
  \[
  \text{s.t.} \quad \text{sufficient conditions for } y \in S(x)
  \]

Difficulty of solving lower-level \( y \)-problems
Overview of methods covered in this tutorial

\[ \min_{x \in \mathcal{X}} F(x) \]
with \( F(x) := \min_{y \in S(x)} f(x, y) \)

 Nested optimization

\[ \min_{x \in \mathcal{X}, y \in \mathcal{Y}} f(x, y) \]
s.t. sufficient conditions for \( y \in S(x) \)

 Constrained optimization

**Difficulty of lower-level \( y \)-problems**

- Implicit gradient
- Explicit gradient (Algorithm unrolling)
- Optimality condition
- Penalty method
Solve simple bilevel optimization via implicit gradients

Start from the simple setting: no constraints, unique lower-level solution
What is the key challenge? Finding implicit gradients

The upper-level gradient w.r.t. $\mathbf{x}$ is

$$
\nabla F(x) = \nabla_x f(x, y^*(x)) + \nabla_x y^*(x) \nabla_y f(x, y^*(x))
$$

**Implicit gradient:** Gradient of the lower-level solution w.r.t. upper-level variable

$$
\nabla_y g(x, y^*(x)) = 0
$$

**Unconstrained + strong convexity**

$$
\nabla^2_{xy} g(x, y^*(x)) + \nabla_x y^*(x) \nabla^2_{yy} g(x, y^*(x)) = 0
$$

$$
\nabla_x y^*(x) := -\nabla^2_{xy} g(x, y^*(x)) \left[\nabla^2_{yy} g(x, y^*(x))\right]^{-1}
$$
Approximate upper-level implicit gradients

Key challenges: evaluating upper-level gradients is costly

\[ \nabla F(x) = \nabla_x f(x, y^*(x)) - \nabla_{xy}^2 g(x, y^*(x)) \left[ \nabla_{yy}^2 g(x, y^*(x)) \right]^{-1} \nabla_y f(x, y^*(x)) \]

Approximate \( y \approx y^*(x) \) and introduce a \textbf{slightly biased} implicit gradient

\[ \overline{\nabla} f(x, y) := \nabla_x f(x, y) + \nabla_{xy}^2 g(x, y) \left[ \nabla_{yy}^2 g(x, y) \right]^{-1} \nabla_y f(x, y) \approx \nabla F(x) \]

Approximate the Hessian inversion by \((N' \sim \mathcal{U}(0, 1, \ldots, N))\)

Neumann series

\[ \left[ \nabla_{yy}^2 g(x, y) \right]^{-1} \approx \frac{N}{L_g} \prod_{n=1}^{N'} \left( \mathbf{I} - \frac{1}{L_g} \nabla_{yy}^2 g(x, y; \phi^n) \right). \]

The generic template: Alternating implicit SGD

**ALSET:** A unified *Alternating Stochastic gradient descent* method

For \( k = 0, 1, 2, \ldots, K \) do

**S1)** \( x^{k+1} = \text{SGD update} \ (x^k) \text{ on } F(x) \text{ with } y^k \approx y^*(x^k) \)

**S2)** \( y^{k+1} = \text{One or multiple SGD updates} \ (y^k) \text{ on } g(x^{k+1}, y) \)

- Reduce to stochastic gradient descent ascent methods [Jin-Netrapalli-Jordan 2019]
Error induced from inexact lower-level variables

**ALSET**: Use inexact lower-level solution to calculate implicit gradient $\nabla F(x)$

**Challenge**: Gradient bias depends on the drift of lower-level solutions $y^*(x)$
Two early attempts to this problem

- **Two-timescale**: Update $x$ in a *slower* timescale than $y$; e.g., TTSA [Hong et al, 20]

  $x^k \rightarrow x^{k+1} \rightarrow x^{k+2} \rightarrow x^{k+3} \rightarrow x^{k+T}$

  $y^k \rightarrow y^{k+1} \rightarrow y^{k+2} \rightarrow y^{k+3} \rightarrow y^{k+T}$

  $\approx y^* (x^{k+T})$

- **Double-loop**: Update $y$ with *growing* # of iters; BSA [Ghadimi et al, 18], StocBio [Ji et al, 21]

  $x^k \rightarrow x^{k+1} \rightarrow x^{k+2} \rightarrow x^{k+3} \rightarrow x^{k+T}$

  $y^k \rightarrow y^{k+1} \rightarrow y^{k+2} \rightarrow y^{k+3} \rightarrow y^{k+T}$

  $\approx y^* (x^{k+T})$

Hong, Wai, Wang, and Yang, “A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic.” *SIAM J OPT 2023*


Ji, Yang, and Liang, “Bilevel optimization: Convergence analysis and enhanced design.” *ICML 2021*
Demystify alternating SGD for bilevel problems

Q: Something not **uncovered** by these analysis?

A1: Update of $x$ uses **decaying stepsizes** $\alpha_k$ to cancel noise; it is **slow**!

A2: The lower-level solution is **highly smooth**; its drift $O(\alpha_k^2)$ is small!

Existing two-timescale/double-loop analysis does not capture this …
Solving (a class of) bilevel problems with SGD convergence rate!

Chen, Sun, and Yin, "Closing the Gap: Tighter Analysis of Alternating Stochastic Gradient Methods for Bilevel Problems" NeurIPS 2021

A1) upper objective $f(x, y)$ and its gradient are Lipschitz continuous
A2) lower objective $g(x, y)$ is strongly convex and smooth in $y$
A3) stochastic 1st- and 2nd-order information are unbiased w/ bounded variance

Theorem (Convergence)

Under the above assumption, if we choose stepsizes $\alpha_k = \mathcal{O}(K^{-\frac{1}{2}})$ and $\beta_k = \mathcal{O}(K^{-\frac{1}{2}})$, without inner loop and increasing batchsize, ALSET satisfies

Upper level
\[
\frac{1}{K} \sum_{k=1}^{K} \mathbb{E} \left[ \| \nabla F(x^K) \|^2 \right] = \mathcal{O} \left( \frac{1}{\sqrt{K}} \right)
\]

Lower level
\[
\mathbb{E} \left[ \| y^K - y^*(x^K) \|^2 \right] = \mathcal{O} \left( \frac{1}{\sqrt{K}} \right)
\]

SGD-like guarantee for certain bilevel problems
SGD-like guarantee for certain nested problems

<table>
<thead>
<tr>
<th>problem class</th>
<th># of loops</th>
<th>batch size</th>
<th>sample complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALSET</td>
<td>Bilevel</td>
<td>Single</td>
<td>$O(\epsilon^{-2})$</td>
</tr>
<tr>
<td>ALSET</td>
<td>Min-max</td>
<td>Single</td>
<td>$O(\epsilon^{-2})$</td>
</tr>
<tr>
<td>ALSET</td>
<td>Compositional</td>
<td>Single</td>
<td>$O(\epsilon^{-2})$</td>
</tr>
<tr>
<td>SGD</td>
<td>Single-level</td>
<td>Single</td>
<td>$O(\epsilon^{-2})$</td>
</tr>
</tbody>
</table>

Sample complexity to achieve an $\epsilon$-stationary point of $F(x)$; i.e., $\mathbb{E}[\|\nabla F(x)\|^2] \leq \epsilon$.

Do not be afraid of solving certain bilevel problems!
Empirical benefits in meta learning

- Meta learning for multiple sinusoidal regression tasks
- **Bilevel** SGD-based ALSET versus **standard MAML** and other bilevel baselines

*Chen, Sun, Yin, “Solving stochastic compositional optimization is nearly as easy as solving stochastic optimization,” *IEEE TSP*, 2021*
Other recent implicit gradient methods not covered

- **Acceleration methods for implicit gradient methods**
  [Khanduri et al., 2021], [Yang et al., 2021], [Shen and Chen, 2022], [Li et al., 2022], [Ji et al., 2022], [Huang et al., 2022], [Dagréou et al., 2022], [Chen et al., 2023], [Khanduri et al., 2023], etc.

**SUSTAIN**: add momentum in upper- and lower-level updates

For $k = 0, 1, 2, \ldots, K$ do

S1) $x^{k+1} = \text{momentum SGD} \ (x^k, y^k) \ \text{on} \ F(x)$

S2) $y^{k+1} = \text{momentum SGD} \ (x^{k+1}, y^k) \ \text{on} \ g(x^{k+1}, y)$

**SUSTAIN** achieves $\mathcal{O}(\epsilon^{-1.5})$ iteration complexity which is near-optimal


Yang, Ji, and Liang, "Provably faster algorithms for bilevel optimization," *Proc. NeurIPS 2021*
Overview of methods covered in this tutorial

**Nested optimization**

\[
\min_{x \in \mathcal{X}} F(x) \\
\text{with } F(x) := \min_{y \in \mathcal{S}(x)} f(x, y)
\]

**Constrained optimization**

\[
\min_{x \in \mathcal{X}, y \in \mathcal{Y}} f(x, y) \\
\text{s.t. } \text{sufficient conditions for } y \in \mathcal{S}(x)
\]

**Difficulty of lower-level y-problems**

- **Implicit gradient**
- **Explicit gradient (Algorithm unrolling)**

Work as SGD when implicit function is smooth, but what if it is not?
How to apply bilevel to more challenging settings?

Extend to …

Non-strongly convex lower-level problems
Overview of methods covered in this tutorial

\[ \min_{x \in \mathcal{X}} F(x) \]
with \[ F(x) := \min_{y \in S(x)} f(x, y) \]

Nested optimization

\[ \min_{x \in \mathcal{X}, y \in \mathcal{Y}} f(x, y) \]
s.t. sufficient conditions for \( y \in S(x) \)

Constrained optimization

Difficulty of lower-level \( y \)-problems

Implicit gradient

Explicit gradient

Optimality condition

Penalty method

Limited applicability; Great when it works

Simple to program; Use approximations
Challenges due to non-convexity

\[
\begin{align*}
\min_{x \in \mathbb{R}^d} & \quad F(x) := f(x, y^*(x)) \\
\text{s.t.} & \quad y^*(x) = \arg \min_{y \in \mathbb{R}^{d'}} g(x, y) 
\end{align*}
\]

(upper level)

(lower level)

Non-unique solutions

Non-differentiable upper-level loss; non-invertible Hessian

\[
\begin{align*}
\min_{x \in \mathcal{X}, y \in \mathcal{Y}} & \quad f(x, y) \\
\text{s.t.} & \quad \text{sufficient conditions for } y \in S(x)
\end{align*}
\]

Constrained reformulation
Main assumption: Polyak-Łojasiewicz (PL) condition

Loss of over-parametrized model is non-convex but satisfies the PL-inequality:

\[ \| \nabla_y g(x, y) \|^2 \gtrsim g(x, y) - \min_y g(x, y) \]

All stationary points are global optimal solutions…

Liu, Zhu, and Belkin, “Loss landscapes and optimization in over-parameterized non-linear systems and neural networks” *Applied and Computational Harmonic Analysis*
Constrained reformulation

\[
\min_{x,y} f(x, y)
\]

\text{s. t. equivalent condition of LL optimality}

KKT conditions

Constraint qualification (CQ) conditions:
Ensure KKT conditions are necessary optimality conditions.
Constrained reformulation

### Value-function based

\[
\min_{x, y \in \mathbb{R}^d} \quad f(x, y) \\
\text{s.t.} \quad g(x, y) - \min_{y' \in \mathbb{R}^d} g(x, y') = 0
\]

### Gradient based

\[
\min_{x, y \in \mathbb{R}^d} \quad f(x, y) \\
\text{s.t.} \quad \nabla_y g(x, y) = 0
\]

Gradient based KKT conditions hold at global optimal set!
Can we formally justify this phenomenon?

— Identify the CQ satisfied by PL bilevel problems
Calmness condition

**Definition (Calmness CQ)**

Let \((x^*, y^*)\) be the global optimal point of

\[
\min_{x,y} f(x, y) \quad \text{s.t.} \quad h(x, y) = 0.
\]

If there exists \(\epsilon\) and \(M\) s.t.

- for any \(\|q\| \leq \epsilon\)
- \(\|(x', y') - (x^*, y^*)\| \leq \epsilon\)

which satisfies \(h(x', y') + q = 0\), one has

\[
f(x', y') - f(x^*, y^*) + M\|q\| \geq 0
\]

then the problem is said to be calm.

- Quantifies the **sensitivity of the objective to the perturbation on constraints.**
- **Key observation:** gradient based PL bilevel problems inherit the calmness CQ!
New necessary condition

Theorem (Necessary condition of KKT)

If $g(x, \cdot)$ satisfies the PL condition and is smooth, and $f(x, \cdot)$ is Lipschitz continuous, then there exists $w^* \neq 0$ such that

$$
R_x(x^*, y^*, w^*) := \| \nabla_x f(x^*, y^*) + \nabla_{xy}^2 g(x^*, y^*) w^* \|^2 = 0
$$

$$
R_w(x^*, y^*, w^*) := \| \nabla_{yy}^2 g(x^*, y^*) (\nabla_y f(x^*, y^*) + \nabla_{yy}^2 g(x^*, y^*) w^*) \|^2 = 0
$$

$$
R_y(x^*, y^*) := \| \nabla_y g(x^*, y^*) \|^2 = 0
$$

hold at the global minimizer $(x^*, y^*)$ of the PL bilevel problem.

- **Compare with KKT: Shadow implicit gradient:**

$$
w^*(x, y) \in \arg \min_w \mathcal{L}(x, y; w) := \frac{1}{2} \| \nabla_y f(x, y) + \nabla_{yy}^2 g(x, y) w \|^2
$$
A generalized alternating gradient method

Using fixed-point equation and the alternating strategy:

For $k = 0, 1, 2, \ldots, K$ do

S1) $y^{k+1} = \text{One or multiple GD updates } (y^k) \text{ on } g(x^k, y)$

S2) $w^{k+1} = \text{GD updates } (w^k) \text{ on } L(x^k, y^{k+1}, w)$

S3) $x^{k+1} = x^k - \alpha (\nabla_{x} f(x^k, y^{k+1}) + \nabla_{xy}^2 g(x^k, y^{k+1}) w^{k+1})$

**GALET :** Generalized ALternating mEthod for bilevel opTimization

Convergence results: GD-like guarantee

A1) upper objective $f(x, y)$ and its gradient are Lipschitz continuous
A2) lower objective $g(x, y)$ is PL, smooth and Hessian-Lipschitz in $y$
A3) The nonzero eigenvalue of the Hessian of $g(x, y)$ is bounded away from 0

Theorem (Convergence)

Under the above assumptions, if we choose stepsizes properly, the iterates generated by the GALET satisfies

Upper-level \[ \frac{1}{K} \sum_{k=0}^{K-1} R_x(x^k, y^k, w^k) = O \left( \frac{1}{K} \right) \]

Lower-level \[ \frac{1}{K} \sum_{k=0}^{K-1} R_y(x^k, y^k) = O \left( \frac{1}{K} \right) \]

Shadow implicit gradient level \[ \frac{1}{K} \sum_{k=0}^{K-1} R_w(x^k, y^k, w^k) = O \left( \frac{1}{K} \right) \]

GALET enjoys the same convergence rate as GD!

Overview of methods covered in this tutorial

Nested optimization

\[
\min_{x \in \mathcal{X}} F(x)
\]
with \( F(x) := \min_{y \in S(x)} f(x, y) \)

Constrained optimization

\[
\min_{x \in \mathcal{X}, y \in \mathcal{Y}} f(x, y)
\]
s.t. sufficient conditions for \( y \in S(x) \)

Difficulty of solving lower-level \( y \)-problems

- Implicit gradient
- Explicit gradient
- Optimality condition
- Penalty method

Limited applicability; Great when it works
Penalty-based reformulations

Under the PL condition, both of the following functions are optimality metrics.

\[ p(x, y) = g(x, y) - v(x) \quad \text{with} \quad v(x) := \min_y g(x, y) \]

\[ p(x, y) = \| \nabla_y g(x, y) \|^2 \]

\[
\begin{align*}
\min_{x \in \mathcal{X}, y \in \mathcal{Y}} \quad & f(x, y) \\
\text{s.t.} \quad & \text{sufficient condition : } p(x, y) \leq 0
\end{align*}
\]

Constrained reformulation
Constrained versus penalized reformulations

Consider a slightly relaxed version of bilevel problem:

\[ \mathcal{BP}_\epsilon : \min_{x,y} f(x, y) \quad \text{s.t. } p(x, y) \leq \epsilon \]

\[ \mathcal{BP}_{\gamma_p} : \min_{x,y} f(x, y) + \gamma p(x, y) \]

Equivalence? Equivalence: all local and global solutions match…

Image from depositphotos.com
Difficulty in preserving local solutions

$$\mathcal{BP}_{\epsilon=0} : \min_{x,y \in \mathbb{R}} \sin^2 \left( y - \frac{2\pi}{3} \right) \quad \text{s.t.} \quad \| \nabla_y g(x,y) \|^2 = (y + \sin(2y))^2 \leq 0$$

Solution is 0

Penalize with gradient norm

$$\mathcal{BP}_{\gamma\rho} : \min \sin^2 \left( y - \frac{2\pi}{3} \right) + \gamma (y + \sin(2y))^2$$

Conditions of preserving local solutions

\[ \mathcal{BP}_\epsilon : \min_{x,y} f(x, y) \quad \text{s.t.} \quad p(x, y) \leq \epsilon \]

Equivalence?

\[ \mathcal{BP}_{\gamma p} : \min_{x,y} f(x, y) + \gamma p(x, y) \]

**Theorem (equivalence)**

Given any \( \epsilon > 0 \), choose the penalty constant \( \gamma \gtrsim \epsilon^{-0.5} \).

i) For \( p(x, y) = g(x, y) - v(x) \), no further assumption is needed;

ii) For \( p(x, y) = \| \nabla_y g(x, y) \|^2 \), further assume the singular values >0;

Then any local solution of the penalized problem \( \mathcal{BP}_{\gamma p} \) is a local solution of the \( \epsilon \)-approximate original bilevel problem \( \mathcal{BP}_\epsilon \).
An alternative method: Penalty-based gradient descent

\[
\min_{x,y} F_\gamma(x, y) := f(x, y) + \gamma(g(x, y) - v(x)) \quad \text{with} \quad v(x) := \min_y g(x, y)
\]

Gradient of value function is computed by a generalized Daskin's theorem:

\[
\nabla_x F_\gamma(x, y) = \nabla_x f(x, y) + \gamma(\nabla_x g(x, y) - \nabla_x g(x, y^*)) , \quad y^* \in \text{arg min } g(x, y)
\]

For \( k = 0, 1, 2, \ldots, K \) do

\begin{align*}
\text{S1)} \quad & x_{k+1} = x_k - \alpha \left( \nabla_x f(x_k, y_k) + \gamma \left( \nabla_x g(x_k, y_k) - \nabla_x g(x_k, \hat{y}_{kT+1}^+) \right) \right) \\
\text{S2)} \quad & y_{k+1} = y_k - \alpha \left( \nabla_y f(x_k, y_k) + \gamma \nabla_y g(x_k, y_k) \right)
\end{align*}

- One only needs first-order derivatives!
Training efficiency for nonconvex bilevel problems

\[ \min_{x,y} F_\gamma(x, y) := f(x, y) + \gamma(g(x, y) - v(x)) \text{ with } v(x) := \min_y g(x, y) \]

**Theorem (convergence)**

Consider running V-PBGD for \( k = 1, 2, \ldots, K \). With small enough step sizes and \( T_k \gtrsim \log k \), it holds that

\[ \frac{1}{K} \sum_{k=1}^{K} \| \nabla F_\gamma(x_k, y_k) \|^2 = O \left( \frac{\gamma}{K} \right) \]

- With \( \gamma \gtrsim \epsilon^{-0.5} \), it implies the \( O(\epsilon^{-1.5}) \) iteration complexity
Overview of methods covered in this tutorial

1. **Implicit gradient**
   - Limited applicability;
   - Great when it works;
   - Incur approximations

2. **Explicit gradient**
   - Simple to program;
   - Incur approximations

3. **Optimality condition**
   - Limited applicability;
   - Great when it works

4. **Penalty method**
   - Broad applicability;
   - Often require relaxations

**Nested optimization**

\[
\min_{x \in \mathcal{X}} F(x)
\]

with

\[
F(x) := \min_{y \in S(x)} f(x, y)
\]

**Constrained optimization**

\[
\min_{x \in \mathcal{X}, y \in \mathcal{Y}} f(x, y)
\]

s.t. sufficient conditions for \( y \in S(x) \)
Other recent advances not covered

- **Acceleration methods for implicit gradient methods**
  [Khanduri et al., 2021], [Yang et al., 2021], [Shen and Chen, 2022], [Li et al., 2022], [Ji et al., 2022], [Huang et al., 2022], [Dagréou et al., 2022], [Chen et al., 2023], [Khanduri et al., 2023], etc

- **Memory-efficient variants for algorithm unrolling methods**
  [Maclaurin et al., 2015], [Pedregosa 2016], [Franceschi et al., 2017, 2018], [Nichol et al., 2018], [Shaban et al., 2019], [Grazzi et al., 2020], [Liu et al., 2021], [Liu et al., 2022], [Bolte et al., 2022]

- **Penalty and primal-dual methods for bilevel optimization**
  [Ye et al., 1997], [Lin et al., 2014], [Liu et al., 2021], [Mehra and Hamm, 2021], [Sow et al., 2022], [Gao et al., 2022], [Ye et al., 2022], [Lu and Mei 2023], [Huang 2023], [Kwon et al., 2023], etc
Simulation: Data hyper-cleaning

In data hyper-cleaning, we try to clean up the polluted training data.

\[ D_{tr} \text{ has polluted data} \quad D_{val} \text{ is clean} \]

Want to learn an importance weight for each data

\[ \omega_i(x), d_i \in D_{tr} \]

Given weights, the models fit the weighted data

\[ \sum_{d_i \in D_{tr}} \omega_i(x) f_{ce}(y; d_i) - \min_y \sum_{d_i \in D_{tr}} \omega_i(x) f_{ce}(y; d_i) \leq \epsilon \]
Simulation: Data hyper-cleaning

We want such models to fit well with clean data:

\[
\min_{x,y} \sum_{d_i \in D_{val}} f_{ce}(y;d_i) \quad \text{s.t.} \quad \sum_{d_i \in D_{tr}} \omega_i(x) f_{ce}(y;d_i) - \min_y \sum_{d_i \in D_{tr}} \omega_i(x) f_{ce}(y;d_i) \leq \epsilon
\]

We evaluate all algorithms with three main metrics:

- **Test accuracy**: classification accuracy of \( y \)
- **F1 score**: precision and recall of cleaner \( x \)
- **Scalability**: Peak GPU memory usage through training and inference
Simulation: Data hyper-cleaning

<table>
<thead>
<tr>
<th>Method</th>
<th>Linear model</th>
<th>2-layer MLP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Test accuracy</td>
<td>F1 score</td>
</tr>
<tr>
<td>RHG</td>
<td>87.64 ± 0.19</td>
<td>89.71 ± 0.25</td>
</tr>
<tr>
<td>T-RHG</td>
<td>87.63 ± 0.19</td>
<td>89.04 ± 0.24</td>
</tr>
<tr>
<td>BOME</td>
<td>87.09 ± 0.14</td>
<td>89.83 ± 0.18</td>
</tr>
<tr>
<td>G-PBGD</td>
<td>90.09 ± 0.12</td>
<td>90.82 ± 0.19</td>
</tr>
<tr>
<td>IAPTT-GM</td>
<td>90.44 ± 0.14</td>
<td>91.89 ± 0.15</td>
</tr>
<tr>
<td>V-PBGD</td>
<td>90.48 ± 0.13</td>
<td>91.99 ± 0.14</td>
</tr>
</tbody>
</table>

- V-PBGD does not have as large memory increase, thanks to being first-order.
Outline

- Part I - Introduction and background
- Part II – Bilevel optimization fundamentals
- Part III – Bilevel applications to reinforcement learning (60 mins)
- Part IV – Multi-objective learning beyond bilevel optimization
- Part V - Conclusions and open directions
Tutorial Part III:
Bilevel applications to reinforcement learning

Zhuoran Yang

Yale Statistics and Data Science

February 20, 2024
Empirical successes of reinforcement learning

AlphaGo

Rubik’s cube

Computer games

Finetune LLMs
Supervised learning

Collect data, train model, and make predictions with the model
Single-agent reinforcement learning

**Single-agent RL:** One agent takes an action $a_h$ at each step $h$

- Environment state $s_{h+1}$ evolves according to agent’s action $a_h$
- Goal: maximize the cumulative rewards $\sum_{h=1}^{H} r(s_h, a_h)$
- Solution concept: optimal policy $\pi^*$ – reward maximizing policy
Multi-agent reinforcement learning

Multi-agent RL: Multiple agents, each takes an action at each step
• Environment state evolves according to actions of all agents
• Multi-objective optimization:
  • each agent aims to maximize her own cumulative rewards
• Game-theoretic solution concepts:
  • Markov perfect equilibria, Coarse correlated equilibria, ...
Bilevel optimization meets reinforcement learning

**Bilevel RL:** Multi-agent RL + leader-follower structure

**Upper-level variable** \( x = \text{policy of leader} \)

\[
\max_{x \in \mathcal{X}, y} f(x, y) \quad \text{(upper level)}
\]

\[ y \in \arg \max_{y' \in \mathcal{Y}} g(x, y') \quad \text{(lower level)} \]

**Lower-level variable** \( y = \text{policy of follower} \)

**Leader’s problem**

**Follower’s problem**

Leader and follower have different reward functions

\[
f(x, y) = \mathbb{E}_{x,y} [\sum_{h=1}^{H} R(s_h, a_h, b_h)]
\]

\[
g(x, y) = \mathbb{E}_{x,y} [\sum_{h=1}^{H} r(s_h, a_h, b_h)]
\]

\( f(x, y) \) and \( g(x, y) \) are cumulative rewards of leader and follower

Hierarchical structure – follower’s problem as leader’s constraint
Interpretation of bilevel RL

Nested in RL problem with parameter $x$

RL problem with policy $x$

Policy $x$

feedback $r_x$

model

data

policy $\pi_y$
Interpretation of bilevel RL

\[
\begin{align*}
\max_{x \in X, y} & \quad f(x, y) \quad \text{(upper level)} \\
\text{s.t.} & \quad y \in \arg \max_{y' \in Y} g(x, y') \quad \text{(lower level)}
\end{align*}
\]

- **Leader** announces a policy \( x \), promise she will play \( x \)
- **Follower** decides his policy \( y \) – best-response to \( x \)
- Leader and follower then play \((x, y)\) simultaneously
- A sequence of state-action-rewards are generated
- Leader and follower receive \( f(x, y) \) and \( g(x, y) \) in total

**Upper**: Find leader’s optimal policy

**Lower**: follower always adopts best response

Leader

Follower

Announce \( x \)

Choose \( y = S(x) \)
Interpretation of bilevel RL

\[
\begin{align*}
\max_{x \in \mathcal{X}, y} & \quad f(x, y) & \text{ (upper level)} \\
\text{s.t.} & \quad y \in \arg \max_{y' \in \mathcal{Y}} g(x, y') & \text{ (lower level)}
\end{align*}
\]

Upper: Find leader’s optimal policy
Lower: follower always adopts best response

Follower’s best response: \( y = S(x) \in \arg \max_{y' \in \mathcal{Y}} g(x, y') \)
Leader’s optimization: \( \max_{x \in \mathcal{X}} f(x, S(x)) \)
Optimal solution pair: \((x^*, S(x^*)) = \text{Stackelberg equil.}\)

\( x^* \) is leader’s optimal policy given that follower responds optimally
A more general view of bilevel RL

More generally, we can have **multiple leaders** \((n)\) and **followers** \((m)\)

Upper-level variable \(\{x^1, \ldots, x^m\} = \text{policies of leaders} \)

Solve \(\text{Equil}\left(\{f^i(x^1, \ldots, x^m, y^1, \ldots, y^n), i = 1, \ldots, m\}\right)\)

\(\text{s.t.} \quad (y^1, \ldots, y^n) \in \text{Equil}\left(\{g^j(x^1, \ldots, x^m, y^1, \ldots, y^n), j = 1, \ldots, n\}\right)\)

Lower-level variable \(\{y^1, \ldots, y^n\} = \text{policies of followers} \)

**Leaders** announce their policies and promise to commit to them

**Followers** form an equilibrium induced by leaders’ policies

Each leader’s goal: steer the system in her favor (game of leaders)
Example: Stackelberg game

\[
\begin{align*}
\min_{x \in \mathcal{X}, y} & \quad f(x, y) \quad \text{(upper level)} \\
\text{s.t.} & \quad y \in \arg \min_{y' \in \mathcal{Y}} g(x, y') \quad \text{(lower level)}
\end{align*}
\]

Upper: Find leader’s optimal policy

Lower: follower always adopts best response

Matrix game with action spaces \( \mathcal{A} = \{a_1, \ldots, a_m\} \), \( \mathcal{B} = \{b_1, \ldots, b_n\} \)

Reward functions \( R(a, b), r(a, b) \) \quad Policies \( x \in \Delta_m, y \in \Delta_n \)

\[
\begin{align*}
f(x, y) &= \mathbb{E}_{a \sim x, b \sim y}[R(a, b)], \quad g(x, y) = \mathbb{E}_{a \sim x, b \sim y}[r(a, b)]
\end{align*}
\]

Best response \( S(x) = \delta\{\arg \max_{b \in \mathcal{B}} \mathbb{E}_{a \sim x}[r(a, b)]\} \)

Follower labels leader’s \( x \) using a deterministic function

Example: Stackelberg game with quantal response

$$\begin{align*}
\min_{x \in \mathcal{X}, y} & \quad f(x, y) & \text{(upper level)} \\
\text{s.t.} & \quad y \in \arg \min_{y' \in \mathcal{Y}} g(x, y') & \text{(lower level)}
\end{align*}$$

Upper: Find leader’s optimal policy

Lower: follower always adopts best response

Matrix game with action spaces $\mathcal{A} = \{a_1, \ldots, a_m\}$, $\mathcal{B} = \{b_1, \ldots, b_n\}$

Reward functions $R(a, b)$, $r(a, b)$ Policies $x \in \mathcal{X} = \Delta_m$, $y \in \mathcal{Y} = \Delta_n$

$$f(x, y) = \mathbb{E}_{a \sim x, b \sim y}[R(a, b)], \quad g(x, y) = \mathbb{E}_{a \sim x, b \sim y}[r(a, b)] + \eta^{-1} \cdot \mathcal{H}(y) \iff \text{entropy}$$

Quantal response $S(x)(b) = Z(x)^{-1} \cdot \exp(\eta \cdot r(x, b))$ Stochastic response


Example: contract design

\[
\begin{align*}
\min_{x \in \mathcal{X}, y} & \quad f(x, y) \quad \text{(upper level)} \\
\text{s.t.} & \quad y \in \arg\min_{y' \in \mathcal{Y}} g(x, y') \quad \text{(lower level)}
\end{align*}
\]

Upper: Find leader’s optimal contract
Lower: follower always adopts best response

Follower takes action \( b \) which generates an outcome \( o \sim p_b \)

Each action \( b \) requires some effort and thus incurs a cost \( c(b) \)

Leader incentivizes follower with an outcome-dependent payment \( S(o) \)

Special case of Stackelberg game with structured rewards

\[
R(S, b) = \mathbb{E}_{o \sim p_b} [R(o) - S(o)], \quad r(S, b) = \mathbb{E}_{o \sim p_b} [S(o)] - c(b)
\]

Example: performative prediction

\[
\begin{align*}
\min_{x \in \mathcal{X}, y} & \quad f(x, y) \quad & \text{(upper level)} \\
\text{s.t.} & \quad y \in \arg \min_{y' \in \mathcal{Y}} g(x, y') \quad & \text{(lower level)}
\end{align*}
\]

Upper: Find leader’s optimal decision

Lower: sample \( z \sim \mathcal{D}(x) \)

\( x \): parameter of a ML model

\( \ell(x, z) \): loss of model \( x \) on data \( z \)

\[
\min_x L(x) = \mathbb{E}_{z \sim \mathcal{D}(x)}[\ell(x, z)]
\]

\( S(x) = \mathcal{D}(x) \): strategically manipulated distribution

Example of \( \mathcal{D}(x) \):

\[
z = Ax + \zeta \iff S(x) = \arg \min_y \text{KL}(y \parallel \mathcal{N}(Ax, \sigma^2 I))
\]
Example: multi-agent performative game

\[ x = (x^1, \ldots, x^n): \text{decision variables of } n \text{ agents} \]

\[ L^i(x^1, \ldots, x^n) = \mathbb{E}_{(z^1, \ldots, z^n) \sim D(x^1, \ldots, x^n)}[\ell(x^i, z^i)] \]

\( D(x) \): decision-dependent observations
\( z^i \) depends on actions of all \( n \) agents

- Example of \( D(x) \):
  \[ z^i = A^i x^i + \sum_{j \neq i} B^{ij} x^j + \zeta^i \]

- Solution concept: Nash equilibrium \( x^* \)

\[ x^{i,*} \in \arg \min_{x^i} L^i(x^i, x^{-i,*}) \]

Upper: Find leader’s equilibrium policy
Lower: sample \( z \sim D(x) \)

Piliouras, Georgios, and Fang-Yi Yu. “Multi-agent performative prediction: From global stability and optimality to chaos.” *EC* 2023
1. Collect bunch of trajectory pairs with policy
2. Humans label the preferred one; Reward predictor predicts the trajectory label
3. Train reward predictor with MLE loss given human labels (Bradley-Terry model)
4. Train policy to increase the predicted reward

Data: two trajectories $\tau_1, \tau_2 \sim \pi_y$, label $z$ given by human

$$\max_{x,y} \mathbb{E}_{\tau_1, \tau_2 \sim \pi_y, z} [z \cdot \log \mathbb{P}_x (\tau_1 > \tau_2) + (1 - z) \cdot \mathbb{P}_x (\tau_1 < \tau_2)] \quad r_x \text{ is MLE}$$

$$\text{s.t.} \quad y \in \arg \max g(x, y') = \mathbb{E}_{\tau \sim \pi_y'} [\sum_{h=1}^H r_x (s_h, a_h)] \quad \pi_y \text{ is optimal wrt } r_x$$

**Example: RLHF / reward design / inverse RL**

$$\min_{x \in \mathcal{X}, y} f(x, y) \quad \text{(upper level)}$$

s.t. $$y \in \arg\min_{y' \in \mathcal{Y}} g(x, y') \quad \text{(lower level)}$$

**Upper:** Find leader’s optimal reward param.

**Lower:** follower always adopts optimal policy

---

**Leader** chooses a reward $$r_x$$  
**Follower** chooses a policy $$\pi_y$$

Leader’s goal: find a reward $$r^*$$ such that

Trajectory $$\tau$$ generated by $$\pi^*$$ explains observed data

$$f(x, y) = \mathbb{E}_{\tau \sim \pi_y, \bar{\tau} \sim \text{Data}}[\text{Dist}(\tau, \bar{\tau})]$$

$$g(x, y) = \mathbb{E}_{\tau \sim \pi_y} [\sum_{h=1}^{H} r_x(s_h, a_h)]$$

Often $$\pi_y$$ does not enter $$f$$ or $$g$$ directly, rather indirectly through $$\tau$$

$$\pi_y \longrightarrow \tau = \{s_h, a_h\}_{h=1}^{H} \longrightarrow g(x, y) \& f(x, y)$$
**Agenda: Recent optimization and learning results**

Upper-level variable $x = \text{policy of leader}$

$$\max_{x \in X, y} f(x, y) \quad \text{(upper level)}$$

s.t. $y \in \arg \max_{y' \in Y} g(x, y') \quad \text{(lower level)}$

Lower-level variable $y = \text{policy of follower}$

**Upper:** Find leader's optimal policy

**Lower:** follower always adopts best response

**Optimization:** When model is known, how to compute $x^*$ and $S(x^*)$?

**Learning (Statistics):** How to learn $(x^*, S(x^*))$ from data efficiently? What data? How many data points needed?
Optimization in bilevel RL – main takeaways

\[
\begin{align*}
\min_{x \in \mathcal{X}, y} & \quad f(x, y) \quad \text{(upper level)} \\
\text{s.t.} & \quad y \in \arg\min_{y' \in \mathcal{Y}} g(x, y') \quad \text{(lower level)}
\end{align*}
\]

- Lower problem is convex optimization (\(y\) is not a policy), rather easy to solve using standard optimization tools
- Lower problem is RL (\(y\) is a policy \(\pi_y\)), need to modify bilevel optimization tools (e.g., penalty method)
Lower problem is not RL – Stackelberg matrix game

Matrix game with action spaces $\mathcal{A} = \{a_1, \ldots, a_m\}$, $\mathcal{B} = \{b_1, \ldots, b_n\}$

Reward functions $R(a, b), r(a, b)$ Policies $x \in \mathcal{X} = \Delta_m, y \in \mathcal{Y} = \Delta_n$

$$f(x, y) = \mathbb{E}_{a \sim x, b \sim y}[R(a, b)], \quad g(x, y) = \mathbb{E}_{a \sim x, b \sim y}[r(a, b)]$$

Solve by LP – find the optimal $x$ for each $b \in \mathcal{B}$

$$\mathcal{X}(b) = \{x \in \Delta_m : y^*(x) = \delta_b\} = \{x : \mathbb{E}_{a \sim x}[r(a, b)] \geq \mathbb{E}_{a \sim x}[r(a, b')], \forall b' \in \mathcal{B}\}$$

$$x^*_b = \arg \max_{x \in \mathcal{X}(b)} \mathbb{E}_{a \sim x}[R(a, b)], \quad \forall b \in \mathcal{B}$$

Enumerate all $b \in \mathcal{B}$ to get solution: $b^* \in \arg \max_b \mathbb{E}_{a \sim x^*_b}[R(a, b)]$

Quantal Stackelberg matrix game

Matrix game with action spaces $\mathcal{A} = \{a_1, \ldots, a_m\}$, $\mathcal{B} = \{b_1, \ldots, b_n\}$

Reward functions $R(a, b), r(a, b)$

Policies $x \in \mathcal{X} = \Delta_m$, $y \in \mathcal{Y} = \Delta_n$

$f(x, y) = \mathbb{E}_{a \sim x, b \sim y}[R(a, b)]$, $g(x, y) = \mathbb{E}_{a \sim x, b \sim y}[r(a, b)] + \eta^{-1} \cdot \mathcal{H}(y) \Leftarrow \text{entropy}$

Quantal response $S(x)(b) = Z(x)^{-1} \cdot \exp(\eta \cdot r(x, b))$

Plug in closed-form of $S(x)$ — reduce to nonlinear optimization:

$$\max_{x \in \mathcal{X}} F(x) = \mathbb{E}_{a \sim x, b \sim S(x)}[R(a, b)]$$

Can be solved by first-order optimization when $\mathcal{B}$ finite
Closed-from of $S(x) + \text{policy gradient}$

Quantal response $S(x)(b) = Z(x)^{-1} \cdot \exp(\eta \cdot r(x, b))$

$$\max_{x \in \mathcal{X}} F(x) = \mathbb{E}_{a \sim x, b \sim S(x)}[R(a, b)]$$

Policy gradient trick: $\nabla_x (\mathbb{E}_{b \sim P_x}[h(b)]) = \mathbb{E}_{b \sim P_x}[h(b) \cdot \nabla_x \log P_x(b)]$

$S(x) \Rightarrow P_x$

Policy gradient trick: $\nabla_x (\mathbb{E}_{b \sim P_x}[h(b)]) = \mathbb{E}_{b \sim P_x}[h(b) \cdot \nabla_x \log P_x(b)]$

---

**Performative prediction** can also be solved by first-order optimization:

$$\nabla_x L(x) = \nabla_x \mathbb{E}_{z \sim D(x)}[\ell(x, z)] = \mathbb{E}_{z \sim D(x)}[\nabla_x \ell(x, z) + \ell(x, z) \cdot \nabla_x \log D(x)]$$


Miller, John P., Juan C. Perdomo, and Tijana Zrnic. "Outside the echo chamber: Optimizing the performative risk." *ICML* 2021
Lower problem is RL – reward design

\[
\begin{align*}
\min_{x,y} f(x, y) \\
\text{s.t. } \pi_y \in S(x) = \arg \max_{y'} g(x, y') = \mathbb{E}_{\tau \sim \pi_y} \left[ \sum_{h=1}^{H} r_x(s_h, a_h) \right]
\end{align*}
\]

Challenge of RL: optimal policy nonunique
Lower problem not convex

Typically lower-level function is \textbf{strongly convex} so that

\[
\min_{x} f(x, S(x)) \quad \frac{\partial f(x, S(x))}{\partial x} = \nabla_x f(x, S(x)) + \nabla S(x) \nabla_y f(x, S(x))
\]

Given by Implicit function theorem

Difficult to apply existing bilevel optimization algorithms directly
Ensure unique lower-level solution – regularization

\[
g(x, y) = \mathbb{E}_{\tau \sim \pi_y} \left[ \sum_{h=1}^{H} \{ r_x(s_h, a_h) + \eta \cdot \mathcal{H}(\pi_y(\cdot | s_h)) \} \right] \quad (\eta > 0)
\]

\[
S(x) = \arg\max_{\pi_y} g(x, y) \text{ unique} \quad \mathcal{H}(p) = \sum_{a \in A} -p(a) \log p(a)
\]
Recall: two general recipes for bilevel optimization

\[
\min_{x \in \mathcal{X}, \ y} \ f(x, y) \\
\text{s.t.} \quad y \in S(x) := \arg\min_{y' \in \mathcal{Y}} \ g(x, y')
\]

\[
\min_{x \in \mathcal{X}} \ F(x) \\
\text{with} \quad F(x) := \min_{y \in S(x)} \ f(x, y)
\]

Nested optimization first over \( y \) and then over \( x \)

\[
\min_{x \in \mathcal{X}, \ y \in \mathcal{Y}} \ f(x, y) \\
\text{s.t.} \quad \text{sufficient conditions for} \ y \in S(x)
\]

Constrained optimization jointly over \( x \) and \( y \)

Implicit gradient

How to compute implicit gradient?

Penalty method

What penalty function? How is regularized problem related to original problem?
Implicit gradient for bilevel RL

Assume leader’s objective depends on $x$ and $\pi_y$ via a bivariate function $U$

$$f(x, y) = \mathbb{E}_{\tau \sim \pi_y} \left[ \sum_{h=1}^{H} U(s_h, a_h; x) \right] \quad S(x) = \arg\max_y g(x, y)$$

Apply policy gradient theorem to $\nabla_x F(x) = \nabla_x f(x, S(x))$

$$\nabla_x f(x, S(x)) = \mathbb{E}_{\tau \sim \pi_{S(x)}} \left[ \sum_{h=1}^{H} \nabla_x U(s_h, a_h; x) \right]$$

$$+ \mathbb{E}_{\pi_x \sim \pi_{S(x)}} \left[ \sum_{h=1}^{H} U(s_h, a_h; x) \cdot \nabla_x \log \pi_{S(x)}(a_h|s_h) \right]$$

Second term contains implicit gradient (apply chain rule):

$$\nabla_x \log \pi_{S(x)}(a|s) = \left[ \nabla_x S(x) \right] (\nabla_y \log \pi_y(a|s)) \bigg|_{y=S(x)}$$

Compute implicit gradient by differentiate lower level

Apply policy gradient theorem to $\nabla_x F(x) = \nabla_x f(x, S(x))$

$$\nabla_x f(x, S(x)) = \mathbb{E}_{\tau \sim \pi_{S(x)}} \left[ \sum_{h=1}^{H} \nabla_x U(s_h, a_h; x) \right]$$

$$+ \mathbb{E}_{\pi_x \sim \pi_{S(x)}} \left[ \sum_{h=1}^{H} U(s_h, a_h; x) \cdot \nabla_x \log \pi_{S(x)}(a_h | s_h) \right]$$

Second term contains implicit gradient (apply chain rule):

$$\nabla_x \log \pi_{S(x)}(a | s) = [\nabla_x S(x)](\nabla_y \log \pi_y(a | s)) \bigg|_{y=S(x)}$$

How to compute $\nabla_x S(x)$? Again, differentiate lower level optimality condition:

$$\nabla_y g(x, S(x)) = 0 \quad \forall x.$$ 

$$\implies \nabla_{xy} g(x, S(x)) + [\nabla_x S(x)] \nabla_{yy} g(x, S(x)) = 0$$

Implicit gradient formula

Apply policy gradient theorem to \( \nabla_x F(x) = \nabla_x f(x, S(x)) \)

\[
\nabla_x f(x, S(x)) = \mathbb{E}_{\tau \sim \pi_{S(x)}} \left[ \sum_{h=1}^{H} \nabla_x U(s_h, a_h; x) \right] \\
+ \mathbb{E}_{\pi_x \sim \pi_{S(x)}} \left[ \sum_{h=1}^{H} U(s_h, a_h; x) \cdot \nabla_x \log \pi_{S(x)}(a_h | s_h) \right]
\]

Second term contains implicit gradient (apply chain rule):

\[
\nabla_x \log \pi_{S(x)}(a | s) = [\nabla_x S(x)](\nabla_y \log \pi_y(a | s)) \bigg|_{y=S(x)}
\]

Implicit gradient formula:

\[
\nabla_x \log \pi_{S(x)}(a | s) = -[\nabla_{xy} g(x, S(x))] [\nabla_{yy} g(x, S(x))]^{-1} [\nabla_y \log \pi_y(a | s)] \bigg|_{y=S(x)}
\]

Note: require policy Hessian \( \nabla_{yy} g(x, y) \)

Recall: penalty method for bilevel optimization

\[
\begin{align*}
\min_{x \in \mathcal{X}, y} & \quad f(x, y) \\
\text{s.t.} & \quad y \in S(x) = \arg \max_y g(x, y)
\end{align*}
\]

Note: lower problem changed to max

Example in part II: \( p(x, y) = \max_{y'} g(x, y') - g(x, y) \)

Question: How to define \( p(x, y) \) for \( \pi_y \)?

Is \( \mathcal{BP}_{\gamma P} \) equivalent to \( \mathcal{BP}_\epsilon \) for some \( \epsilon \)?

\[
\begin{align*}
\mathcal{BP}_{\gamma P} : & \quad \min_{x, y} f(x, y) + \gamma p(x, y) \\
\mathcal{BP}_\epsilon : & \quad \min_{x, y} f(x, y) \quad \text{s.t.} \; p(x, y) \leq \epsilon
\end{align*}
\]
Penalty function I – value penalty

\[ g(x, y) = \mathbb{E}_{\tau \sim \pi_y} \left[ \sum_{h=1}^{H} r(x(s_h, a_h)) \right] \]

\[ p(x, y) = \max_{y'} g(x, y') - g(x, y) \]

Note: optimal value \( \max_{y'} g(x, y') = g(x, S(x)) \) is unique
\( S(x) \) might be non-unique

**Theorem (solution relation)**
Assume \( f(x, \cdot) \) is \( L \)-Lipschitz in \( y \). For any \( \epsilon > 0 \), choosing \( \lambda = \mathcal{O}(L/\epsilon) \), any local/global solution to \( \mathcal{BP}_{\gamma, p} \) is a local/global solution to \( \mathcal{BP}_\epsilon \).

No explicit regularization required. Uniqueness not necessary.

Penalty function I – value penalty

* solve a RL problem → \( S(x) \)

* policy evaluation with vector reward \( \nabla_x r_x \)

**Gradient of \( p(x, y) \)**

\[
\nabla_x p(x, y) = -\nabla_x g(x, y) + \nabla_x g(x, y) \bigg|_{y=S(x)} \\
= \mathbb{E}_{\tau \sim S(x)} \left[ \sum_{h=1}^{H} \nabla x r_x (s_h, a_h) \right] - \mathbb{E}_{\tau \sim \pi_y} \left[ \sum_{h=1}^{H} \nabla x r_x (s_h, a_h) \right] \\
\nabla_y p(x, y) = -\nabla_y g(x, y) = \text{policy gradient}
\]

\[\mathcal{BP}_{\gamma_p} : \min_{x, y} f(x, y) + \gamma p(x, y)\]

\[\mathcal{BP}_\epsilon : \min_{x, y} f(x, y) \quad \text{s.t.} \quad p(x, y) \leq \epsilon\]

- optimality of \((x_\lambda, y_\lambda)\) in \(\mathcal{BP}_\lambda\)
- monotonicity at \((x_\lambda, y_\lambda)\):
  \[
  \langle \nabla_y p(x_\lambda, y_\lambda), y - y_\lambda \rangle \geq C \cdot p(x_\lambda, y_\lambda), \quad \forall y
  \]
Penalty function 2 – Bellman penalty

\[ g(x, y) = \mathbb{E}_{\tau \sim \pi_y} \left[ \sum_{h=1}^{H} \{ r(x, s_h, a_h) + \eta \cdot \mathcal{H}(\pi_y(\cdot | s_h)) \} \right] \quad (\eta > 0) \]

Optimal policy \( \pi^* = S(x) \) characterized by optimal \( Q \) function \( Q^*_x(s, a) \):

\[ \pi^* = \arg \max_{\pi_y} \mathbb{E}_{s \sim \rho} \left[ \sum_{a \in A} \pi_y(a | s) \cdot Q^*_x(s, a) \right] + \eta \cdot \mathcal{H}(\pi_y(\cdot | s)) \]

\[ = h(x, y) \]

\[ p(x, y) = \max_{y'} h(x, y') - h(x, y) \]

**Theorem (solution relation)**

Assume \( f(x, \cdot) \) is \( L \)-Lipschitz in \( y \). For any \( \epsilon > 0 \), choosing \( \gamma = \mathcal{O}(\sqrt{L\eta^{-1}\epsilon^{-1}}) \), any local/global solution to \( \mathcal{BP}_{\gamma, p} \) is a local/global solution to \( \mathcal{BP}_\epsilon \).
Penalty function 2 – Bellman penalty

\[ p(x, y) = \max_{y'} h(x, y') - h(x, y) \]

\[ h(x, y) = \mathbb{E}_{s \sim \rho} \left[ \sum_{a \in A} \pi_y(a|s) \cdot Q_x^*(s, a) + \eta \cdot H(\pi_y(\cdot|s)) \right] \]

**Gradient of \( p(x, y) \)**

\[ \nabla_x p(x, y) = -\nabla_x h(x, y) + \nabla_x h(x, y) \bigg|_{y = S(x)} \]

\[ = \mathbb{E}_{(s,a) \sim S(x)} \left[ \nabla_x Q_x^*(s_h, a_h) \right] - \mathbb{E}_{(s,a) \sim \pi_y} \left[ \nabla_x Q_x^*(s_h, a_h) \right] \]

\[ \nabla_y p(x, y) = -\nabla_y h(x, y) = \text{policy gradient} \]
Implement penalty method for bilevel RL

**PBRL algorithm**

At current iterate \((x_k, y_k)\)

- Solve the MDP with reward \(r_{x_k}\) and get \(\pi^k \approx S(x_k)\) or \(Q^k \approx Q^*_k\).

- Use \(\pi^k\) (version 1) or \(Q^k\) (version 2) to approximate \(\nabla p(x_k, y_k)\).

- Get gradient \(\nabla f(x_k, y_k) + \lambda \nabla p(x_k, y_k)\).

- Update \((x^{k+1}, y^{k+1})\) via (policy) gradient methods.

First order updates – do not require policy Hessian

Inner MDP solving subroutine – linear convergence using policy mirror descent

Outer loop Converge to a stationary point at sublinear rate

Numerical experiments: RLHF on Atari games

The Arcade learning environment is commonly used to test RL algorithms

- **Goal**: finish the games with high score
- **Input**: sequence of images
- **Output**: actions to play

The OpenAI gymnasium library includes 59 games
We implement

- **Baseline**: original RLHF algorithm (DRLHF)
- **Ours**: PBRL algorithm
- **Oracle**: A2C with access to the ground truth reward

We follow the original RLHF paper and use the game score as the ground truth reward and generate human feedback.
Numerical experiments: RLHF on Atari games

MsPacman

BeamRider

Seaquest
Online learning in bilevel RL – setting

\[
\begin{align*}
\min_{x \in \mathcal{X}, y} & \quad f(x, y) \\
\text{s.t.} & \quad y \in S(x) := \arg \min_{y' \in \mathcal{Y}} g(x, y')
\end{align*}
\]

\[
\begin{align*}
\min_x & \quad F(x) \\
\text{with} & \quad F(x) = f(x, S(x))
\end{align*}
\]

Leader’s learning problem: learns \(x^*\) from data by interacting with follower
\[
\text{regret}_K = \sum_{k=1}^K [F(x^*) - F(x^k)]
\]

Unknowns: \(f(x, y)\) and \(S(x)\)

Main challenge: estimate \(S(x)\)

Assumptions on data & model?

Zhao, Geng, Banghua Zhu, Jiantao Jiao, and Michael Jordan. "Online learning in stackelberg games with an omniscient follower." ICML 2023
Online learning in Stackelberg game

Matrix game with action spaces $\mathcal{A} = \{a_1, \ldots, a_m\}, \mathcal{B} = \{b_1, \ldots, b_n\}$

Reward functions $R(a, b), r(a, b)$ Policies $x \in \mathcal{X} = \Delta_m, y \in \mathcal{Y} = \Delta_n$

$f(x, y) = \mathbb{E}_{a \sim x, b \sim y}[R(a, b)], \quad g(x, y) = \mathbb{E}_{a \sim x, b \sim y}[r(a, b)]$

Best response $S(x) = \delta\{\arg\max_{b \in \mathcal{B}} \mathbb{E}_{a \sim x}[r(a, b)]\}$

Data assumption: Learner controls both players
Toy example: online learning in Stackelberg game

Data assumption: Learner controls both players, observes bandit feedbacks of \((R, r)\)

Algorithm:
- Try all \((a, b) \in A \times B\) for \(N\) times, estimate \(\hat{R}\) and \(\hat{r}\)
- Return \(\hat{x} = \arg \max_x f(x, \hat{S}(x))\)

A pessimistic result:
- No matter how accurate \(\hat{R}\) and \(\hat{r}\) are, \(\hat{x}\) can be worse than \(x^*\) by a constant
- \(\hat{S}(x) = \delta\{\arg \max_{b \in B} E_{a \sim x}[\hat{r}(a, b)]\}\) is sensitive to estimation error

\[\implies \text{Best respose } S(x) \text{ cannot be estimated by estimating } r\]
Method I – forget about estimating $S(x)$

Data assumption: Learner controls leader, observes bandit feedbacks of $F(x)$
Leader play $a \sim x$, follower plays $b \sim S(x)$, receive $R(a, b)$

Algorithm:
- discretize $\mathcal{X}$ by $\mathcal{E}_\epsilon(\mathcal{X})$
- treat each $x \in \mathcal{E}_\epsilon(\mathcal{X})$ as an arm and run UCB algorithm

Theorem (Zhu et al)

For contract design with $m$ possible outcomes, the regret is $\tilde{O}(K^{1-1/(m+2)})$.
That is, to find $\epsilon$-optimal solution, we need $\mathcal{O}((1/\epsilon)^{m+2})$ samples.

Method II – estimate $S(x)$ via quantal response

**Data assumption:** Learner controls leader, observes bandit feedbacks and follower’s action. Leader plays $a \sim x$, follower plays $b \sim S(x)$, receive $R(a, b)$

\[
f(x, y) = \mathbb{E}_{a \sim x, b \sim y}[R(a, b)], \quad g(x, y) = \mathbb{E}_{a \sim x, b \sim y}[r(a, b)] + \eta^{-1} \cdot \mathcal{H}(y) \iff \text{entropy}
\]

Quantal response $S(x)(b) = Z(x)^{-1} \cdot \exp(\eta \cdot r(x, b))$

**Algorithm:** estimate $r$ via MLE + UCB bonus

- Estimate $r$ from MLE $\hat{r} = \arg \max \log \mathbb{P}_r(b \mid x)$
- Estimate $R$ by mean estimation $\hat{R}$
- UCB planning: $\max_x \langle \hat{R} + \Gamma_1, x \times S_\hat{r}(x) \rangle + \Gamma_2(x)$

**Theorem**

\[
\text{regret}_K = \tilde{O}(\sqrt{T})
\]

Summary

- Bilevel RL – Leader-follower structure + RL
- Examples – Stackelberg game, RLHF / reward design
- Optimization aspect of bilevel RL
  - Implicit gradient
  - Penalty method
- Learning aspect of bilevel RL
  - UCB + discretization
  - Quantal response + MLE + UCB
Outline

- Part I - Introduction and background
- Part II – Bilevel optimization fundamentals
- Part III – Bilevel applications to reinforcement learning
- Part IV – Multi-objective learning beyond bilevel optimization (65 mins)
- Part V - Conclusions and open directions
Tutorial Part IV:
Multi-objective Learning Beyond Bilevel

Lisha Chen

Rensselaer Polytechnic Institute

February 20, 2024
Outline

- Introduction and motivation
  - Motivation
  - Solution concepts and measures of optimality
- Multi-gradient based methods
  - (deterministic) MGDA, CAGrad, other methods
  - (stochastic) SMG, MoCo, MoDo
- Theory of multi-objective learning
  - Optimization
  - Generalization
- Application of multi-objective learning
Success of AI in the new era

To merge two dictionaries in Python, you can use the `update()` method.
Tasks, data, metrics all can be modeled as an objective...

$$\min_{\theta} \text{loss (model } \theta, \text{ training data, metric, tasks)}$$
Tackling multiple tasks, data, metrics via single-objective learning ... 

\[
\begin{align*}
\min_{\theta} \text{loss (model }, & \text{ )} \\
+ \\
\min_{\theta} \text{loss (model }, & \text{ )} \\
+ \\
\min_{\theta} \text{loss (model }, & \text{ )}
\end{align*}
\]

Simple but may cause... unit mismatch or competition
Limitations of the weighted sum method

- Hard to pre-define the weights when the scale of the objectives are unknown

- Some optimal solutions cannot be reached by optimizing the weighted sum objective

- Optimization conflicts: some objectives may not be optimized, or even degraded
Weighted sum cannot obtain some solutions

Example:

\[ l_1(\theta) = 1 - e^{-\|\theta - \frac{1}{\sqrt{n}}\|_2^2}, \]

\[ l_2(\theta) = 1 - e^{-\|\theta + \frac{1}{\sqrt{n}}\|_2^2}, \]

Cannot find points in the middle of the Pareto front even if change different weights

Debabrata Mahapatra, Vaibhav Rajan ¨Multi-Task Learning with User Preferences: Gradient Descent with Controlled Ascent in Pareto Optimization¨ Proc. ICML 2020
Weighted sum cannot obtain some solutions

Example:

Limitations of the weighted sum method

- Hard to pre-define the weights when the scale of the objectives are unknown

- Some optimal solutions cannot be reached by optimizing the weighted sum objective

- Optimization conflicts: some objectives may not be optimized, or even degraded
Optimization conflicts

Loss landscape of $\ell_1$  
Gradient of $\ell_1$  
Gradient of $\ell_2$  
Loss landscape of $\ell_2$

$\theta > 90^\circ$
Examples of optimization conflicts in large language models

Accuracy objective dominates!

Cannot be solved by merely increasing the model scale or finetuning! [Wei et al. '23]

Need to rethink LLM training with safety objective!
Examples of optimization conflicts in multi-modal learning

Modality competition

Results in suboptimal training errors, thus some modalities are unexplored.

Formulation for multi-objective learning

\[ \min_{\theta} \quad L_S(\theta) = [\ell_1(\theta, S), \ldots, \ell_t(\theta, S), \ldots, \ell_T(\theta, S)] \]

A vector optimization problem

How to optimize a vector?

\[
\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad ? \\
\begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad ?
\]
Partial ordering

A binary relation $\leq$ defined in a real linear space $R^T$ that satisfies the following axioms (for arbitrary $w, x, y, z \in R^T$):

- Reflexive: $x \leq x$;
- Transitive: $x \leq y, y \leq z \Rightarrow x \leq z$;
- $x \leq y, w \leq z \Rightarrow x + w \leq y + z$;
- $x \leq y, \alpha \in R_+ \Rightarrow \alpha x \leq \alpha y$;
Lexicographical ordering

On $R^T$, a lexicographic order $\leq_{lex}$ is defined in the following manner. Let $x = [x_1, x_2, ..., x_T]^T$ and $y = [y_1, y_2, ..., y_T]^T$ be in $R^T$.

Then $x \leq_{lex} y$ if

(a) $x = y$ or
(b) if $x \neq y$ and $t_0 = \min \{t: x_t \neq y_t\}$, then $x_{t_0} < y_{t_0}$.

The order depends on the order of the first element that differs.
Multi-level optimization induced by lexicographical ordering

\[
\begin{align*}
\min_\theta & \quad \ell_t(\theta), \quad t = 1, 2, \ldots, T \\
\text{s.t.} & \quad \ell_j(\theta) \leq \min_\theta \ell_j(\theta), \quad \text{for all } j = 1, 2, \ldots, t - 1, t > 1
\end{align*}
\]

A simple multi-level optimization problem with one variable \( \theta \)

A simple bilevel optimization problem with one variable \( \theta \) and when \( T = 2 \)
Epsilon-constraint methods

Idea: optimize one objective conditioned on that the rest objectives are within pre-defined thresholds

\[
\min_\theta \quad \ell_T(\theta) \\
\text{s.t.} \quad \ell_j(\theta) - \min_\theta \ell_j(\theta) \leq \epsilon_j, \quad \text{for all } j = 1, 2, \ldots, T - 1
\]

Can find different points on the Pareto front corresponding to different trade-offs/preferences
Natural ordering

A component-wise partial ordering, denoted as $\leq_C$

Natural ordering cone: $C := \{ x \in \mathbb{R}^T \mid 0 \leq x \}$

$\leq_C := \{ (x, y) \in \mathbb{R}^T \times \mathbb{R}^T \mid y - x \in C \}$
Definition (Pareto optimal)
A point $\theta^* \in \Theta$ is Pareto optimal iff there exists no other point $\theta \in \Theta$ that $L(\theta) \leq L(\theta^*)$, and $\ell_t(\theta) < \ell_t(\theta^*)$ for at least one $t \in [T]$. 

Hypothesis $\mathcal{H}$

Pareto front
Pareto optimality induced by natural ordering

Definition (Pareto optimal)
A point $\theta^* \in \Theta$ is Pareto optimal iff there exists no other point $\theta \in \Theta$ that $L(\theta) \leq_{\mathcal{C}} L(\theta^*)$, and $\ell_t(\theta) < \ell_t(\theta^*)$ for at least one $t \in [T]$.

---

Definition (Pareto stationary) [Fliege et al’ 2020]
A point $\theta^* \in \Theta$ is Pareto stationary iff
\[
\min_{\lambda \in \Delta^T} ||\nabla L(\theta)\lambda||^2 = 0.
\]
Equivalently, $\theta^*$ is Pareto stationary iff there exists no first-order common descent directions for all objectives, i.e.
\[
\text{range} (\nabla L(\theta)) \cap -R^T_+ = \emptyset
\]
Pareto optimality

How to find Pareto optimal/stationary models?

Use scalarization to convert the vector-valued objective to a scalar-valued objective.
Challenge of conflicting gradient

\[ w_1 \ell_1(\theta) + w_2 \ell_2(\theta) \]

Optimization conflicts still exist!
Challenge of conflicting gradient

\[ w_1 \ell_1(\theta) + w_2 \ell_2(\theta) \]

Potentially hurt the convergence of the training error!
Optimization conflicts – what and how

Optimization conflicts

\[ (\theta) \]

\[ \ell_1(\theta) \quad \ell_2(\theta) \]

\[ \langle \nabla \ell_1(\theta), \nabla \ell_2(\theta) \rangle < 0 \]

Common gradient descent to mitigate optimization conflicts

Update direction that decrease all objectives

> 90°
Outline

- Introduction and motivation
  - Motivation
  - Solution concepts and measures of optimality
- Multi-gradient based methods
  - (deterministic) MGDA, CAGrad, other methods
  - (stochastic) SMG, MoCo, MoDo
- Theory of multi-objective learning
  - Optimization
  - Generalization
- Application of multi-objective learning
Conflict-avoidant direction

Conflict-avoidant (CA) direction definition: [Fliege ’00, Désidéri ’12]

\[
\min_{\lambda \in \Delta^T} \langle \nabla L_S(\theta) \lambda, -d \rangle
\]

Worst descent amount

Conflict-avoidant direction

Conflict-avoidant (CA) direction definition: [Fliege ’00, Désidéri ’12]

\[
\max_{d \in \mathbb{R}^d} \min_{\lambda \in \Delta^T} \langle \nabla L_S(\theta) \lambda, -d \rangle - \frac{1}{2} \|d\|^2
\]

Worst descent amount

Regularization term

Jean-Antoine Désidéri, „Multiple-gradient Descent Algorithm (MGDA) for Multi-objective Optimization”. Comptes Rendus Mathematique, 350(5-6), 2012.
Conflict-avoidant direction

Reformulation:

\[ d(\theta) = -\nabla L_S(\theta) \lambda^*(\theta) \quad \text{s.t.} \quad \lambda^*(\theta) \in \arg \min_{\lambda \in \Delta^T} \| \nabla L_S(\theta) \lambda \|^2. \]

Idea: each update iteration follows the CA direction with a changing \( \lambda \)

\[ \theta_{k+1} = \theta_k + \alpha d(\theta_k) \]

Multiple gradient descent (MGDA) or dynamic weighting algorithms
A variant of MGDA - CAGrad

Idea: to find a steepest descent direction subject to the constraint that it is close to a prior direction $-g_0$

$$g_0 = \frac{1}{T} \nabla L_S(\theta) \mathbf{1}$$

$$\begin{align*}
\max_{d \in \mathbb{R}^q} \min_{i \in [T]} \langle \nabla \ell_i(\theta), -d \rangle & \quad \text{s.t.} \quad ||d + g_0|| \leq c||g_0||,
\end{align*}$$

Reformulate as

$$d = -(g_0 + \nabla L_S(\theta) \lambda^*(\theta))$$

$$\lambda^*(\theta) = \arg\min_{\lambda \in \Delta^T} \langle \nabla L_S(\theta) \lambda, g_0 \rangle + \frac{1}{\phi^2} ||\nabla L_S(\theta) \lambda||$$

$$\phi = c^2 ||g_0||^2$$

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, Qiang Liu, "Conflict-Averse Gradient Descent for Multi-task Learning," NeurIPS 2021
Other methods for multi-task learning – PCGrad

Idea: to find a combination of the directions that are projections onto the normal plane of their conflicting gradients

\[
d = - \sum_t \nabla \ell_{S,t}(\theta)^{PC}
\]

\[
\nabla \ell_{S,t}(\theta)^{PC} = \nabla \ell_{S,t}(\theta) - \frac{\langle \nabla \ell_{S,t}(\theta), \nabla \ell_{S,j}(\theta) \rangle}{||\nabla \ell_{S,j}(\theta)||^2} \cdot \nabla \ell_{S,j}(\theta)
\]
Other methods for multi-task learning – Nash-MTL

Idea: to find a scale-invariant update direction

\[ d(\theta) = -\nabla L_S(\theta) \lambda^*(\theta) \]

Solve \( \lambda^*(\theta) \) that \( \nabla L_S(\theta)^T \nabla L_S(\theta) \lambda^*(\theta) = 1/\lambda^*(\theta) \)

Change the scale of \( L_S(\theta) \) does not change \( d(\theta) \)

Other methods not covered

- **Gradient manipulation / dynamic weighting methods**
  - GradNorm [Chen’ 18]
  - GradDrop [Chen’ 20]
  - IMTL [Liu’ 21]
  - UW [Kendall’ 18]
  - RLW [Lin’ 22]
  - Nash-MTL [Navon’ 22]

- **(Stochastic) MGDA-type methods**
  - CR-MOGM [Zhou’ 22]
  - SDMGrad [Xiao’ 23]

Not an exhaustive list
Good news for MGDA in modern MOL

Multi-Task Learning as Multi-Objective Optimization

Conflict-Averse Gradient Descent for Multi-task Learning

MGDA-type algorithms recently applied to multi-task learning
Sad news for MGDA in modern MOL?

Test performance not as good as static weighting…
MGDA not as expected in modern MOL?
Two root causes of degraded performance

- Optimization / computational

Vanilla stochastic MGDA may not converge to Pareto stationarity.

[Kurin et al. 22']
Two root causes of degraded performance

- **Optimization / computational**

  Vanilla stochastic MGDA may not converge to Pareto stationarity.

- **Generalization / statistical**

  No guarantee that models learned by stochastic MGDA can generalize well.

Test error = optimization error + generalization error

[Inoue et al. 22']

[Kurin et al. 22']
One challenge in stochastic MOL: Bias in updates

Ideal CA direction

\[ \nabla \ell_{S,1}(\theta) \]
\[ \nabla \ell_{S,2}(\theta) \]

Actual stochastic update direction

\[ \nabla \ell_{Z,1}(\theta) \]
\[ \nabla \ell_{Z,2}(\theta) \]

d: a stochastic sample

One challenge in stochastic MOL: Bias in updates

Example with 2 objectives ($T = 2$) and exactly solving subproblems

$$
\lambda^*(\theta) = \left[ \frac{(\nabla \ell_{S,2}(\theta) - \nabla \ell_{S,1}(\theta))^\top \nabla \ell_{S,2}(\theta)}{\|\nabla \ell_{S,1}(\theta) - \nabla \ell_{S,2}(\theta)\|^2} \right]_{[0,1]} \text{ solves } \min_{\lambda \in [0,1]} \|\lambda \nabla \ell_{S,1}(\theta) + (1 - \lambda) \nabla \ell_{S,2}(\theta)\|^2
$$
One challenge in stochastic MOL: Bias in updates

Example with 2 objectives \( T = 2 \) and exactly solving subproblems

\[
\lambda^*(\theta) = \left[ \frac{(\nabla \ell_{S,2}(\theta) - \nabla \ell_{S,1}(\theta))^\top \nabla \ell_{S,2}(\theta)}{\|\nabla \ell_{S,1}(\theta) - \nabla \ell_{S,2}(\theta)\|^2} \right]_{[0,1]} \text{ solves } \min_{\lambda \in [0,1]} \|\lambda \nabla \ell_{S,1}(\theta) + (1 - \lambda) \nabla \ell_{S,2}(\theta)\|^2
\]

\[\neq\]

\[
\lambda^*(\theta, z) = \left[ \frac{(\nabla \ell_{z,2}(\theta) - \nabla \ell_{z,1}(\theta))^\top \nabla \ell_{z,2}(\theta)}{\|\nabla \ell_{z,1}(\theta) - \nabla \ell_{z,2}(\theta)\|^2} \right]_{[0,1]}
\]

\[z: \text{ a stochastic sample}\]
One challenge in stochastic MOL: Bias in updates

**Example** with 2 objectives ($T = 2$) and solving stochastic subproblems

\[
\lambda^*(\theta, z) = \left[ \frac{(\nabla \ell_{z,2}(\theta) - \nabla \ell_{z,1}(\theta))^{\top} \nabla \ell_{z,2}(\theta)}{\| \nabla \ell_{z,1}(\theta) - \nabla \ell_{z,2}(\theta) \|^2} \right]_{[0,1]}
\]

**Bias in CA weight**

\[\mathbb{E}_{z \in S}[\lambda^*(\theta, z)] \neq \lambda^*(\theta) := \arg \min_{\lambda \in \Delta_T} \| \nabla L_S(\theta) \lambda \|^2\]

**Bias in CA direction**

\[\mathbb{E}_{z \in S}[-\nabla L_z(\theta) \lambda^*(\theta, z)] \neq d(\theta)\]

Due to the intrinsic **nonlinearity** of the mapping from $\nabla L_S(\theta)$ to $d(\theta)$
A simple stochastic MOO algorithm - SMG

Mini-batch stochastic multi-objective gradient descent

\[
\text{for } k = 0, \ldots, K - 1 \text{ do}
\]
\[
\begin{align*}
&\text{Compute gradient } \nabla L_{z_k}(\theta_k) \\
&\text{Compute dynamic weight } \lambda_{k+1} = \arg\min_{\lambda \in \Delta} \| \nabla L_{z_k}(\theta_k) \lambda \|^2 \\
&\text{Update model parameter } \theta_{k+1} = \theta_k - \alpha \nabla L_{z_k}(\theta_k) \lambda_{k+1}
\end{align*}
\]
\text{end for}

Increasing the batch size [Liu et al ’21]

Variance reduction mitigates the bias due to the continuity from the mapping of gradient \( \nabla L_S(\theta) \) to the update direction \( d(\theta) \)

Increasing batch size [Liu et al ’ 21]

**New problem:**
Inefficient, if not impossible!
A simple stochastic MOO algorithm - MoCo

MoCo: Multi-objective with gradient correction

\begin{algorithm}
\begin{algorithmic}
\For {$k = 0, \ldots, K - 1$}
\State Compute gradient $\nabla L_{z_k}(\theta_k)$
\State Compute moving average of the gradient $Y_{k+1} = Y_k + \nabla L_{z_k}(\theta_k)$
\State Compute dynamic weight $\lambda_{k+1} = \Pi_\Delta^T(\lambda_k - \gamma Y_k^T Y_k \lambda_k)$
\State Update model parameter $\theta_{k+1} = \theta_k - \alpha Y_{k+1} \lambda_{k+1}$
\EndFor
\end{algorithmic}
\end{algorithm}

Use momentum-based methods [Fernando et al '23]

Variance reduction mitigates the bias due to the continuity from the mapping of gradient $\nabla L_S(\theta)$ to the update direction $d(\theta)$

A simple stochastic MOO algorithm - MoCo

**MoCo: Multi-objective with gradient correction**

for $k = 0, \ldots, K - 1$ do

Compute gradient $\nabla L_{z_k}(\theta_k)$

Compute moving average of the gradient $Y_{k+1} = (1 - \beta_k)Y_k + \beta_k \nabla L_{z_k}(\theta_k)$

Compute dynamic weight $\lambda_{k+1} = \Pi_{\Delta} (\lambda_k - \gamma Y_k^T Y_k \lambda_k)$

Update model parameter $\theta_{k+1} = \theta_k - \alpha Y_{k+1} \lambda_{k+1}$

end for

---

A simple stochastic MOO algorithm - MoDo

MoDo: Multi-objective Double sampling optimization

for $k = 0, ..., K - 1$ do

Compute gradients $\nabla L_{z,k,1}(\theta_k)$, $\nabla L_{z,k,2}(\theta_k)$

Compute dynamic weight $\lambda_{k+1} = \Pi_{\Delta T}(\lambda_k - \gamma \nabla L_{z,k,1}(\theta_k)^T \nabla L_{z,k,2}(\theta_k) \lambda_k)$

Update model parameter $\theta_{k+1} = \theta_k - \alpha \nabla L_{z,k+1,1}(\theta_k) \lambda_{k+1}$

end for

Iterative update of weight $\lambda$ instead of exactly solving it

A simple stochastic MOO algorithm - MoDo

MoDo: Multi-objective Double sampling optimization

\[
\text{for } k = 0, \ldots, K - 1 \text{ do } \\
\text{Compute gradients } \nabla L_{Z_k,1}(\theta_k), \nabla L_{Z_k,2}(\theta_k) \\
\text{Compute dynamic weight } \lambda_{k+1} = \Pi_{\Delta t}(\lambda_k - \gamma \nabla L_{Z_k,1}(\theta_k)^T \nabla L_{Z_k,2}(\theta_k) \lambda_k) \\
\text{Update model parameter } \theta_{k+1} = \theta_k - \alpha \nabla L_{Z_{k+1,1}}(\theta_k) \lambda_{k+1} \\
\text{end for}
\]

Double sampling mitigates the bias due to the sample independence

\[
E_{Z_k} \left[ \nabla L_{Z_k,1}(\theta_k)^T \nabla L_{Z_k,2}(\theta_k) \right] = \nabla L_S(\theta_k)^T \nabla L_S(\theta_k)
\]

A simple stochastic MOO algorithm - MoDo

Double sampling mitigates the bias due to the sample independence

\[ E_{Z_K} \left[ \nabla L_{Z,k,1}(\theta_k) \nabla L_{Z,k,2}(\theta_k) \right] = \nabla L_S(\theta_k) \nabla L_S(\theta_k) \]

Outline

- Introduction and motivation
  - Motivation
  - Solution concepts and measures of optimality
- Multi-gradient based methods
  - (deterministic) MGDA, CAGrad, other methods
  - (stochastic) SMG, MoCo, MoDo
- Theory of multi-objective learning
  - Optimization
  - Generalization
- Application of multi-objective learning
Assess the ability to avoid conflicts

How good is the approximate CA direction?
Measure of optimization conflict avoidance

We use two distances as measure of conflict avoidance (CA) ability.

Measure in terms of **CA direction** \( d_{\lambda}(\theta) = -\nabla L_S(\theta)\lambda \)

\[
\mathcal{E}_{ca}(\theta, d_{\lambda}(\theta)) := \|\mathbb{E}_A[d_{\lambda}(\theta) - d(\theta)]\|^2
\]
Measure of optimization conflict avoidance

Measure by the expected distance to the **CA direction** $d_\lambda(\theta)$:

- Measure by the expected distance to the CA direction $d_\lambda(\theta)$:

**Idea:**

Distance to CA direction

- Approximation error to the minimum descent amount across objectives

  Measures CA ability
Definitions & assumptions

Assumptions

A1. Smoothness
A2. Strong convexity
A3. Lipschitz continuity

- Standard assumptions in optimization and algorithm stability analysis
- Separately analyze general nonconvex (A1 & A3) and strongly convex (A1 & A2) settings
Conflict avoidance analysis

Theorem (CA ability guarantee, informal)

Under mild assumptions (A1 & A3, or A1 & A2), and proper choices of step sizes and batch sizes, the CA distance of SMG, MoCo, MoDo, MoCo+ converge to zero as number of iterations increases.

- Demonstrates the benefit of stochastic MGDA methods over static weighting in CA ability.
Optimization analysis

Theorem (PS optimization error guarantee, informal)

Under mild assumptions, the PS optimization error of SMG, MoCo, MoDo, MoCo+ converge to zero as number of iterations increases.

- Choosing proper step sizes, the convergence rates of PS optimization errors of MoDo and MoCo match the convergence rate of SGD.

Convergence rates are summarized next.
Convergence rates

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Batch size</th>
<th>NC</th>
<th>Lipschitz $\lambda^*(x)$</th>
<th>Bounded function</th>
<th>Opt.</th>
<th>CA dist.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMG (Liu and Vicente, 2021, Thm 5.3)</td>
<td>$\mathcal{O}(t)$</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>$T^{-\frac{1}{8}}$</td>
<td>-</td>
</tr>
<tr>
<td>CR-MOGM (Zhou et al., 2022, Thm 3)</td>
<td>$\mathcal{O}(1)$</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>$T^{-\frac{1}{4}}$</td>
<td>-</td>
</tr>
<tr>
<td>MoCo (Fernando et al., 2023, Thm 2)</td>
<td>$\mathcal{O}(1)$</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>$T^{-\frac{1}{20}}$</td>
<td>$T^{-\frac{1}{8}}$</td>
</tr>
<tr>
<td>MoCo (Fernando et al., 2023, Thm 4)</td>
<td>$\mathcal{O}(1)$</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>$T^{-\frac{1}{4}}$</td>
<td>-</td>
</tr>
<tr>
<td>SMG (Ours, Thms 4.1-4.3)</td>
<td>$\mathcal{O}(t)$</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>$T^{-\frac{1}{8}}$</td>
<td>$T^{-\frac{1}{2}}$</td>
</tr>
<tr>
<td>MoCo (Ours, Thms 4.1-4.3)</td>
<td>$\mathcal{O}(1)$</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>$T^{-\frac{1}{16}}$</td>
<td>$T^{-\frac{1}{4}}$</td>
</tr>
<tr>
<td>MoDo (Ours, Thms 3.1,3.3,3.5)</td>
<td>$\mathcal{O}(1)$</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>$T^{-\frac{1}{4}}$</td>
<td>-</td>
</tr>
<tr>
<td>MoDo (Ours, Thms 3.1,3.3,3.5)</td>
<td>$\mathcal{O}(1)$</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>$T^{-\frac{1}{8}}$</td>
<td>$T^{-\frac{1}{4}}$</td>
</tr>
</tbody>
</table>

A new **unified theoretical framework** to analyze optimization and conflict avoidance with **improved assumptions or convergence rates**.

Beyond optimization challenges

Even when the optimization error is small, the generalization error could be large, thus the test error is large.

[Xin et al. 22′]
Test risk decomposition

A measure of test risk tailored for MOL based on Pareto stationarity.

Pareto stationary (PS) test risk decomposition

\[
\min_{\lambda \in \Delta^T} \| \nabla L(\theta) \lambda \| = \min_{\lambda \in \Delta^T} \| \nabla L_S(\theta) \lambda \| + \min_{\lambda \in \Delta^T} \| \nabla L(\theta) \lambda \| - \min_{\lambda \in \Delta^T} \| \nabla L_S(\theta) \lambda \|
\]

PS population risk $R_{\text{pop}}(\theta)$

PS optimization error $R_{S,\text{opt}}(\theta)$

PS generalization error $R_{S,\text{gen}}(\theta)$

Test error = optimization error + generalization error
Generalization analysis

Theorem (Pareto generalization)

In the nonconvex case, if \( \sup_z \mathbb{E}_A \left[ \| \nabla L_z(A(S)) \|_F^2 \right] \leq G^2 \) for any \( S \) with \( |S| = N \), then the generalization errors of MoCo, MoDo satisfy

\[
\mathbb{E}_{A,S}[R_{S,\text{gen}}(A(S))] = \mathbb{E}_{A,S} \left[ \min_{\lambda \in \Delta^T} \| \nabla L(A(S))\lambda \| - \min_{\lambda \in \Delta^S} \| \nabla L_S(A(S))\lambda \| \right] = \mathcal{O} \left( K^{1/2} N^{-1/2} \right)
\]

- A tight bound (matching lower bound) for nonconvex objective functions

Early stopping regime
Generalization analysis via algorithm stability

MOL uniform stability: bound output change after perturbing the training data by one sample

Definition (MOL uniform stability)

A randomized algorithm $A : Z^N \rightarrow R^d$, is MOL-uniformly stable with $\epsilon$, if for all neighboring datasets $S, S'$, we have

\[
\text{Sensitivity metric } \sup_z \mathbb{E}_A \left[ \| \nabla L_z(A(S)) - \nabla L_z(A(S')) \|_F^2 \right] = \epsilon^2
\]
A close look at algorithm stability

Definition (MOL uniform stability)

A randomized algorithm $A : Z^N \rightarrow R^d$, is MOL-uniformly stable with $\epsilon$, if for all neighboring datasets $S, S'$, we have

**Sensitivity metric**

$$\sup_z \mathbb{E}_A \left[ \left\| \nabla L_z (A(S)) - \nabla L_z (A(S')) \right\|_F^2 \right] = \epsilon^2$$

Generated by dataset $S$

Generated by dataset $S'$

Generated by test data $S''$
Generalization analysis via algorithm stability

\[ \mathbb{E}_{A,S}[R_{S, \text{gen}}(A(S))] \leq \epsilon + O\left(N^{-\frac{1}{2}}\right) \]

**MOL uniform stability**: bound output change after perturbing the training data by one sample

In the general nonconvex case

\[ \epsilon \leq (\text{gradient norm bound}) \times P(\text{perturbed sample is selected during training}) \leq K/N \]
Generalization analysis

Theorem (Pareto generalization w/ strong convexity)

In strongly convex case, with proper choice of step sizes, it holds

\[
E_{A,S}[R_{S,\text{gen}}(A(S))] \begin{cases} 
= \mathcal{O}(N^{-\frac{1}{2}}) & \gamma = O(K^{-1}) \\
= \mathcal{O}(K^{\frac{1}{2}}N^{-\frac{1}{2}}) & \text{larger } \gamma
\end{cases}
\]

- Generalization error does not increase with \( K \) if stepsize \( \gamma \) is small
- Matches the generalization error of single-objective learning
Why mitigating conflict may hurt test risk?

In the strongly convex case

**Generalization**

\[
\mathbb{E}_{A,S}[R_{S,\text{gen}}(A(S))]
\begin{cases}
\mathcal{O}(N^{-\frac{1}{2}}) & \gamma = O(K^{-\frac{1}{2}}) \\
\mathcal{O}(K^{\frac{1}{2}}N^{-\frac{1}{2}}) & \text{Larger } \gamma
\end{cases}
\]

**Conflict avoidance**

\[
\frac{1}{K} \sum_{k=1}^{K} \mathcal{E}_{ca}(\theta_k, \lambda_{k+1}) = \mathcal{O}\left(\frac{1}{\gamma K} + \sqrt{\alpha \gamma}\right)
\]

To control optimization error, we set \(\gamma = O(K^{-\frac{1}{2}})\)

\(\gamma \uparrow\), generalization error \(\uparrow\)

\(\gamma \uparrow\), CA ability \(\uparrow\) (CA error \(\downarrow\))
Why mitigating conflict may hurt test risk?

\[
\mathbb{E}_{A,S}[R_{S,gen}(A(S))]
= \mathcal{O}\left(N^{-\frac{1}{2}}\right) \quad \gamma = O(K^{-1})
= \mathcal{O}\left(K^{\frac{1}{2}}N^{-\frac{1}{2}}\right) \quad \text{Larger } \gamma
\]

\[
\frac{1}{K} \sum_{k=1}^{K} \mathcal{E}_{ca}(\theta_k, \lambda_{k+1}) = \mathcal{O}\left(\frac{1}{\gamma K} + \sqrt{\frac{\alpha}{\gamma}}\right)
\]

Generated by a smaller \( \gamma \)

Generated by a larger \( \gamma \)

Stability

Tracking CA direction
Comparison of the three errors for different methods

A new **unified theoretical framework** to analyze the three errors and theory-guided hyperparameter selection to balance among the three errors.

---

Take home message

A new algorithm that interpolates between static and dynamic weighting with **theory-guided hyperparameters** to balance the trade-off!

A new unified theoretical framework to analyze the three errors.

Figure: Three-way trade-off among optimization, generalization, and conflict avoidance.

- ↓: diminishing in an optimal rate w.r.t. $N$;
- ↑: growing w.r.t. $N$;
- : diminishing w.r.t. $N$, but not in an optimal rate.

<table>
<thead>
<tr>
<th>Region</th>
<th>Opt.</th>
<th>Gen.</th>
<th>Conflict</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>↑</td>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>II</td>
<td>↓</td>
<td>↓</td>
<td>↑</td>
</tr>
<tr>
<td>III</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>IV</td>
<td>↓</td>
<td>↓</td>
<td>↓</td>
</tr>
</tbody>
</table>
Application to multi-domain image classification

Office-home dataset
- 4 domains
- 65 classes/domain
- 70-100 images/class

Art
- Clipart
- Product
- Real-world

Application to multi-domain image classification

Holistic performance metric

$$\Delta A\% = \frac{1}{T} \sum_{t=1}^{T} \frac{(S_{B,t} - S_{A,t})}{S_{B,t}} \times 100$$

Table: classification results on Office-home dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Art</th>
<th>Clipart</th>
<th>Product</th>
<th>Real-world</th>
<th>$\Delta A_{st}% \downarrow$</th>
<th>$\Delta A_{id}% \downarrow$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static (EW)</td>
<td>62.99</td>
<td>76.48</td>
<td>88.45</td>
<td>77.72</td>
<td>0.00</td>
<td>5.02</td>
</tr>
<tr>
<td>MGDA-UB (Lin et al., 2022+)</td>
<td>64.32</td>
<td>75.29</td>
<td>89.72</td>
<td>79.35</td>
<td>-1.02</td>
<td>4.04</td>
</tr>
<tr>
<td>GradNorm (Chen et al., 2018)</td>
<td>65.46</td>
<td>75.29</td>
<td>88.66</td>
<td>78.91</td>
<td>-1.03</td>
<td>4.04</td>
</tr>
<tr>
<td>PCGrad (Yu et al., 2020)</td>
<td>63.94</td>
<td>76.05</td>
<td>88.87</td>
<td>78.27</td>
<td>-0.53</td>
<td>4.51</td>
</tr>
<tr>
<td>CAGrad (Liu et al., 2021+)</td>
<td>63.75</td>
<td>75.94</td>
<td>89.08</td>
<td>78.27</td>
<td>-0.48</td>
<td>4.56</td>
</tr>
<tr>
<td>RGW (Lin et al., 2022+)</td>
<td>65.08</td>
<td>78.65</td>
<td>88.66</td>
<td>79.89</td>
<td>-2.30</td>
<td>2.85</td>
</tr>
<tr>
<td>MoCo (Fernando et al., 2023)</td>
<td>64.14</td>
<td>79.85</td>
<td>89.62</td>
<td>79.57</td>
<td>-2.48</td>
<td>2.68</td>
</tr>
<tr>
<td>MoDo (ours)</td>
<td><strong>66.22</strong></td>
<td><strong>78.22</strong></td>
<td><strong>89.83</strong></td>
<td><strong>80.32</strong></td>
<td><strong>-3.08</strong></td>
<td><strong>2.11</strong></td>
</tr>
</tbody>
</table>

Application to scene understanding

<table>
<thead>
<tr>
<th>Object segmentation</th>
<th>Classification of pixel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth regression</td>
<td>Distance to camera</td>
</tr>
<tr>
<td>Surface normal estimation</td>
<td>Direction of surface normal</td>
</tr>
</tbody>
</table>
Application to scene understanding

Table 4: Segmentation, depth, and surface normal estimation results on NYU-v2 dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Segmentation (Higher Better)</th>
<th>Depth (Lower Better)</th>
<th>Surface Normal</th>
<th>(\Delta A_{\text{stat}}) % ↓</th>
<th>(\Delta A_{\text{indep}}) % ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mIoU</td>
<td>Pix Acc</td>
<td>Abs Err</td>
<td>Rel Err</td>
<td>Mean</td>
</tr>
<tr>
<td>Static (EW)</td>
<td>53.77</td>
<td>75.45</td>
<td>0.3845</td>
<td>0.1605</td>
<td>23.5737</td>
</tr>
<tr>
<td>MGDA-UB [38]</td>
<td>50.42</td>
<td>73.46</td>
<td>0.3834</td>
<td>0.1555</td>
<td>22.7827</td>
</tr>
<tr>
<td>GradNorm [4]</td>
<td>53.58</td>
<td>75.06</td>
<td>0.3931</td>
<td>0.1663</td>
<td>23.4360</td>
</tr>
<tr>
<td>PCGrad [46]</td>
<td>53.70</td>
<td>75.41</td>
<td>0.3903</td>
<td>0.1607</td>
<td>23.4281</td>
</tr>
<tr>
<td>CAGrad [57]</td>
<td>53.12</td>
<td>75.19</td>
<td>0.3871</td>
<td>0.1599</td>
<td>22.5257</td>
</tr>
<tr>
<td>RGW [33]</td>
<td>53.85</td>
<td>75.87</td>
<td>0.3772</td>
<td>0.1562</td>
<td>23.6725</td>
</tr>
<tr>
<td>MoCo [3]</td>
<td>54.05</td>
<td>75.58</td>
<td>0.3812</td>
<td>0.1530</td>
<td>23.3868</td>
</tr>
<tr>
<td>MoDo (ours)</td>
<td>53.37</td>
<td>75.25</td>
<td>0.3739</td>
<td>0.1531</td>
<td>23.2228</td>
</tr>
</tbody>
</table>

- MoDo with balanced tradeoff among three metrics outperforms MGDA and static weighting
Application to speech processing

- Pre-training with unlabeled data.
- Downstream fine-tuning.
Multi-lingual, multitask with unified MOL

Over 7000 languages

Universal language translator

Domain-specific jargon

Security and privacy of data

French: "Bonjour les amis"
Swedish: "Hej kompisar"
Russian: "Привет друзья"
Chinese: “你好朋友”

Hello friends

Bonjor les amis
Hej kompisar
Привет друзья
你好朋友
Joint pretraining & multi-lingual finetuning

CPC: contrastive predictive coding loss
CTC: connectionist temporal classification loss

\[
\min_{\theta, \phi} [\ell_{CPC}(\theta), \ell_{CTC}(\theta, \phi_1), \ldots, \ell_{CTC}(\theta, \phi_M)]
\]
Results on benchmarks

Metric: Word Error Rate (WER)

\[
WER = \frac{I + D + S}{N} \times 100\%
\]

- Insertion (I): #words incorrectly added
- Deletion (D): #words undetected
- Substitution (S): #words substituted
- (N): Total #words in the labeled transcript

Baselines:

- Wac2Vec2: a SOTA model
- FT: Supervised baseline without pretraining
- Two stage (PT+FT): 2-stage pretraining & finetuning (without joint MOL)
- Multi-objective (static): without optimization conflict avoidant update
Results on benchmarks

WER Comparison for ASR

WER Comparison for S2TT
Conclusions

Bi-/multi-level

\[
\min \text{ objective 1}
\]

s.t. \( \min \) objective 2

Theory foundation

Multi-objective

\[
\min (\text{obj 1, obj 2, obj 3})
\]
Take home

Multi-objective and multi-level optimization can flexibly model complex learning tasks and enable exciting applications in AI!

Tianyi Chen  Zhuoran Yang  Lisha Chen

THANK YOU!
References


Jean-Antoine Désidéri, "Multiple-gradient Descent Algorithm (MGDA) for Multi-objective Optimization". Comptes Rendus Mathematique, 350(5-6), 2012.


References

Yunwen Lei, ``Stability and Generalization of Stochastic Optimization with Nonconvex and Nonsmooth Problems,” Proc. COLT, 2023


References


References


