Project Report For S&DS 431/631 Final Project: A Gentle Introduction to Bilevel Optimization

Keyi Li
keyi.y.li@yale.edu
Program of Computational Biology and Bioinformatics
Yale University

Abstract

Bilevel optimization is a specialized framework for solving hierarchical problems where one optimization task depends on the solution of another. This project aims to provide a comprehensive introduction to bilevel optimization, explaining its classical foundations and recent advanced applications in deep learning. In the classical setting, we motivate the discussion of theory with simple real-life application and showcase how established theories such as duality and KKT conditions can be leveraged in bilevel optimization. A brief discussion of problem types like optimistic and pessimistic formulations will also be included. For advanced application in deep learning, we focus on scalable algorithms tailored for large-scale models, such as implicit function methods, gradient unrolling, and value function-based approaches. Emphasis is placed on tractable bilevel problems where the lower-level solution is unique. By combining theoretical insights with real-world examples, this project offers an accessible yet detailed exploration of bilevel optimization for both traditional and modern computational settings.

Keywords: bilevel optimization, duality, KKT condition, signal processing, machine learning, problem reformulation

1 Introduction

Bilevel optimization distinguishes itself from traditional optimization approaches through its hierarchical structure, where one optimization problem is nested within another, creating distinct upper and lower level problems with their own objectives and constraints. Mathematically, it can be formulated in the following form(Beck and Schmidt, 2021):

$$\begin{aligned} \min_{x\in X, y} \quad & F(x,y) \\ \text{s.t.} \quad & G(x,y) \geq 0, \\ & y \in S(x), \end{aligned} \tag{1}$$

where $S(x)$ is the set of optimal solutions to the other nested optimization problem:

$$\begin{aligned} \min_{y\in Y} \quad & f(x,y) \\ \text{s.t.} \quad & g(x,y) \geq 0. \end{aligned} \tag{2}$$

Problem (1) is usually named upper level problem or the leader's problem whereas Problem (2) is termed lower level problem or the follower's problem.

Unlike single-level optimization methods, which seek to optimize a single objective subject to a set of constraints, bilevel optimization must contend with the inherent complexity of two decision-makers acting sequentially, where the upper-level optimizer must account for the optimal response from the lower-level problem. This interacting paradigm introduces significant computational challenges and theoretical intricacies, as the lower-level solution serves as an implicit constraint on the upper-level problem, potentially turning the optimization problem non-convex even when both levels' problems are individually well-behaved (Zhang et al., 2023a).

The study of bilevel optimization dates back to Stackelberg games (Richardson, 1995) and has a deep connection with game theory and other fields of economics. It also finds broad applications in day-to-day life in classical setting, such as pricing, toll setting, market modeling and resource allocation. Recently, research interest has surged in the field of bilevel optimization as success application has been demonstrated in the field of signal processing and machine learning, where bilevel optimization is useful for image reconstruction (Crockett and Fessler, 2022), improving model robustness (Zuo et al., 2021), generalizability (Arjovsky et al., 2020), efficiency (Zhang et al., 2023b) and scalability(Zhang et al., 2023b).

This report presents a comprehensive analysis of bilevel optimization across both classical and contemporary deep learning paradigms. The classical framework builds upon established theoretical foundations, incorporating duality theory and Karush-Kuhn-Tucker (KKT) conditions. In the advanced deep learning context, we examine how bilevel optimization enhances model generalizability, particularly focusing on out-of-distribution prediction and domain adaptation challenges where test and training distributions exhibit significant disparities. Our investigation encompasses three principal optimization methodologies: implicit function techniques, gradient unrolling approaches, and value function-based methods, with particular emphasis on the implicit function framework. To facilitate practical understanding, we provide detailed examinations of three case studies, offering concrete insights into solving and applying bilevel optimization problems.

2 Methods and Analysis

Bilevel optimization problems present unique challenges due to their hierarchical structure. Methods for solving these problems can be broadly categories into two classes as detailed below.

2.1 Classical Setting:

The following discussion follows (Li, 2024) and (Beck and Schmidt, 2021).

In this section, we explore methods to transform these problems into more tractable single-level optimization problems, which can then be solved by well-designed solvers.

2.1.1 Optimal solution to follower's problem in closed form

When the optimal solution to follower's problem is unique and can be expressed as a closed form solution, we can easily substitute the solution into leader's problem and solve the bilevel optimization problem in a standard way. A simple example can be found in Section 3.1.

In cases where no closed from solution exists, two primary approaches have emerged: value function reformulation and KKT-based reformulation.

2.1.2 Value Function Reformulation

One intuitive approach to solving bilevel problems is to characterize the follower's optimal solution through its value function. Consider the following bilevel optimization problem:

$$\begin{aligned} \min_{x \in X, y \in Y} \quad & F(x,y) \\ \text{s.t.} \quad & G(x,y) \leq 0 \\ & y \in S(x) \end{aligned} \tag{3}$$

For each leader's decision x, we define the value function of the follower's problem as:

$$\varphi(x) := \min_{y \in Y} \{ f(x,y) : g(x,y) \leq 0 \} \tag{4}$$

This value function is equivalent to the formulation in (2), and implicitly captures the best objective value the follower can achieve for any given leader decision x. Using this function, we can reformulate the bilevel problem as a single-level optimization:

$$\begin{aligned} \min_{x \in X, y \in Y} \quad & F(x,y) \\ \text{s.t.} \quad & G(x,y) \leq 0 \\ & g(x,y) \leq 0 \\ & f(x,y) \geq \varphi(x) \end{aligned} \tag{5}$$

While this reformulation elegantly transforms the bilevel problem into a single-level problem, it introduces a significant computational challenge: the optimal value function $\varphi : \mathbb{R}^{n_x} \to \mathbb{R}$ is typically nonsmooth and difficult to compute explicitly. This limitation motivates the exploration of alternative reformulation approaches.

2.1.3 KKT-Based Reformulations

When the lower-level problem satisfies certain convexity properties, we can leverage optimality conditions to reformulate the bilevel problem. This approach is particularly powerful for two important cases: LP-LP bilevel problems and problems with convex lower levels.

LP-LP Bilevel Problems Consider a bilevel problem where both levels are linear programs:

$$\begin{aligned} \min_{x,y} \quad & c_x^\top x + c_y^\top y \\ \text{s.t.} \quad & Ax + By \leq a \\ & y \in \arg\min_{\bar{y}} \{ d^\top \bar{y} : Cx + D\bar{y} \leq b \} \end{aligned} \tag{6}$$

For linear programs, the KKT conditions are both necessary and sufficient for optimality. This allows us to replace the lower-level optimization problem with its KKT conditions:

$$\begin{aligned} D^\top \lambda = d, \quad \lambda \geq 0 \quad & \text{(dual feasibility)} \\ Cx + Dy \leq b \quad & \text{(primal feasibility)} \\ \lambda_i(C_i x + D_i y - b_i) = 0 \quad \forall i = 1, \ldots, \ell \quad & \text{(Complementary slackness)} \end{aligned} \tag{7}$$

The dual feasibility condition ensures the existence of valid dual variables, while the complementarity condition enforces that either a constraint is active ($C_i x + D_i y = b_i$) or its corresponding dual variable is zero ($\lambda_i = 0$).

General Convex Lower-Level Problems For problems with convex lower levels that satisfy Slater's constraint qualification:

$$\begin{aligned} \min_{x \in X, y \in Y} \quad & F(x, y) \\ \text{s.t.} \quad & G(x, y) \leq 0 \\ & y \in \arg\min_y \{f(x, y) : g(x, y) \leq 0\} \end{aligned} \tag{8}$$

The KKT reformulation yields:

$$\begin{aligned} \min_{x, y, \lambda} \quad & F(x, y) \\ \text{s.t.} \quad & x \in X \\ & \nabla_y L(x, y, \lambda) = 0 \\ & g(x, y) \leq 0 \\ & \lambda \geq 0 \\ & \lambda^\top g(x, y) = 0 \end{aligned} \tag{9}$$

where $L(x, y, \lambda) = f(x, y) - \sum_{i=1}^{\ell} \lambda_i g_i(x, y)$ is the Lagrangian. The stationarity condition $\nabla_y L(x, y, \lambda) = 0$ ensures that y is a critical point of the Lagrangian.

Computational Challenges While KKT reformulations provide a systematic way to transform bilevel problems into single-level problems, they introduce complementarity constraints. This results in a Mathematical Program with Complementarity Constraints (MPCC) or Mathematical Program with Equilibrium Constraints (MPEC), which are inherently non-convex and challenging to solve.

2.2 Advanced Setting:

The following discussion follows (Zhang et al., 2023a).

The discussion is limited to a subclass of bilevel optimization problem that follows the following constraints: (a) the lower-level is unconstrained (b) the optimzal solution to the lower level is a singleton. (c) the objective functions are differentiable.

More specifically, the problem can be abstractly formulated as

$$\min_\theta f(\theta, \phi(\theta^*)) \quad \textbf{s.t.} \;\; \phi(\theta^*)) = \arg\min g(\theta, \phi) \tag{10}$$

With the assumption that the objective function is differentiable, we can have the following result by applying chain rule:

$$\frac{df(\theta, \phi^*(\theta))}{d\theta} = \nabla_\theta f(\theta, \phi^*(\theta)) + \frac{d\phi^*(\theta)^\top}{d\theta} \nabla_\phi f(\theta, \phi^*(\theta)) \tag{11}$$

2.2.1 Implicit Gradient

The implicit gradient-based (IG) function leverage Implicit Function Theorem to approximate $\frac{d\phi^*(\theta)^\top}{d\theta}$. When the lower level problem is unconstrained, simply applying the first-order stationary condition, we have

$$\nabla_\phi g(\theta, \phi^*(\theta)) = \mathbf{0}. \tag{12}$$

.

Assuming that g() is second-order differentiable, by taking derivative with respect to θ, we have

$$\frac{d\phi^*(\theta)}{d\theta} = -\nabla^2_{\theta,\phi} g(\theta, \phi^*(\theta)) \nabla^2_{\phi,\phi} g(\theta, \phi^*(\theta))^{-1}. \tag{13}$$

It's usually hard to calculate Equation (13) exactly as it involves the calculation of mixed (second-order) partial derivative and the inverse of the Hessian. Different approximation techniques are used by different IG-based algorithms to efficiently approximate (13).

Here is the general workflow of IF-based iterative optimization methods:

For iteration t:

- Given θ_t, obtain an approximate solution of the lower-level problem, denoted as $\widetilde{\phi}(\theta_t)$;
- Based on $\widetilde{\phi}(\theta_t)$, combine (11) and (13), we have

  $$\widetilde{\nabla} f(\theta_t) := \nabla_\theta f(\theta_t, \widetilde{\phi}(\theta_t)) - \widetilde{\nabla}^2_{\theta,\phi} g(\theta_t, \widetilde{\phi}(\theta_t)) \widetilde{\nabla}^2_{\phi,\phi} g(\theta_t, \widetilde{\phi}(\theta_t))^{-1} \nabla_\phi f(\theta_t, \widetilde{\phi}(\theta_t))$$

  Approximation methods will be used to calculate $\widetilde{\nabla}^2_{\phi,\phi} g(\theta_t, \widetilde{\phi}(\theta_t))^{-1}$
- $\widetilde{\nabla} f(\theta_t)$ will be used to update θ_t through gradient descent: $\theta_{t+1} \leftarrow \theta_t - \alpha \widetilde{\nabla} f(\theta_t)$

Conjugate Gradient (Shaban et al., 2019) and WoodFisher(Singh and Alistarh, 2020) are usually used to approximate (13).

Conjugate Gradient (CG) For notation simplicity and demonstration purpose, we denote $\widetilde{\nabla}^2_{\phi,\phi} g(\theta_t, \widetilde{\phi}(\theta_t))^{-1} \nabla_\phi f(\theta_t, \widetilde{\phi}(\theta_t))$ as $\mathbf{H}^{-1}\mathbf{g}$.

For CG, $\mathbf{H}^{-1}\mathbf{g}$ is estimated by solving a quadratic program :

$$\min_x \frac{1}{2} x^T \mathbf{H} x - g^T x$$

A typical first-order gradient descent algorithm can be used to solve for $\mathbf{H}^{-1}\mathbf{g}$. However, the convergence rate of CG depends on the smallest singular value of Hessian. An ill-conditioned lower level problem can result in slow convergence.

WoodFisher :

WoodFisher approximation use low-rank approximation to simplify the estimation of the Hessian. It further uses Woodfisher matrix identity to compute matrix inversion.

Take rank-one approximation as an example: Hessian is approximated as

$$\mathbf{H} \approx \mathbf{v}\mathbf{v}^T + \gamma\mathbf{I}$$

where γ can be viewed as a damping term to ensure the invertibility of Hessian.

Using Woodfisher matrix identity, we have

$$\mathbf{H}^{-1}\mathbf{v} \approx \gamma^{-1}\mathbf{v} + \frac{\gamma^{-2}\mathbf{v}\mathbf{v}^\top}{1+\gamma^{-1}\mathbf{v}^\top\mathbf{v}}$$

2.2.2 Gradient Unrolling (GU)

Unlike IF-based methods, GU-based methods uses an intermediary phase: an unrolled lower-level optimizer, to bridges the gap between the lower-level solution and upper-level optimization. Through this structure, automatic differentiation (AD) is used to calculate the gradients with respect to the upper-level optimization variable θ.

Specifically, the GU-based approach approximates the lower-level optimal solution $\phi^*(\theta)$ by running a given algorithm for a fixed number of iterations, and then inserting the entire trajectory into the upper-level objective.

As an abstract example, for the t^{th} iteration, the K step optimal solution is defined as $\widetilde{\phi}(\theta_t) := \phi_K = q(\theta_t, q(\theta_t, \cdots, q(\theta_t, \phi_0)))$, and will be passed to upper level objective. AD will then be used to calculate the gradient for upper-level problem

$$\widetilde{\nabla} f(\theta_t, \widetilde{\phi}(\theta_t)) := \frac{df(\theta_t, q(\theta_t, q(\theta_t, \cdots, q(\theta_t, \phi_0))))}{d\theta} \tag{14}$$

(14) will be used by standard gradient descent algorithm to update θ_t.

2.2.3 Value Function

So far, we talked about IF and GU-based methods for solving bilevel optimization problems that involve "simple" lower-level problems. In more general settings where the lower-level problem is constrained or contains a set of optimal solutions rather than a singleton, value function based methods together with well-developed solver can be used to optimize bilevel objectives. However, these methods are rarely used in practical application because they cannot scale to large models. Thus, in this review, they are categorized as methods for classical setting. A detailed introduction to this class of methods can be found at Section 2.1.

3 Case study and Application

3.1 Case 1 Market Optimization : Reformulation using closed form solution

The following example is adapted from (Li, 2024).

Suppose market price of product is inverse related to the total quantity of products. For simplicity, the relationship is assumed to be $P(Q) = a - b \cdot Q$ where $a, b > 0$. In the following

derivation, we always assume that the leader company moves first and the follower company responds optimally. Let q_l, q_f be the product quantity generated by leader company and follower company. Let c_l, c_f denote the per-unit production cost for leading company and follower company. What's more, both companies want to maximize their own revenue.

Given the above information, the optimization can be formulated as

$$\max_{x_l} \pi_l = \max_{x_l} \left(a - b\left(q_l + q_f^*\right)\right) q_l - c_l \cdot q_l \tag{15}$$

s.t.

$$q_f^* = \arg\max_{q_f} \pi_f = \arg\max_{q_f} \left(a - b\left(q_l + q_f\right)\right) q_f - c_f \cdot q_f \tag{16}$$

It's easy to notice that (16) can be solved in close form, where we set $\frac{\partial \pi_f}{\partial q_f} = 0$, we have

$$q_f^* = \frac{a - c_f - b \cdot q_l}{2b} \tag{17}$$

Substitute (17) into (15), we have

$$\arg\max_{q_f} \pi_l = \arg\max_{q_f} (a-b) \left(q_l + \frac{a - c_f - b \cdot q_l}{2b}\right) q_l - c_l \cdot q_l \tag{18}$$

which is turned into a single-level optimization problem, which can be solved efficiently.

As as summary, we see that when the lower level optimization problem can give closed form solution, the bilevel optimization can be turned into single-level optimization problem, which will be much easier to solve. However, in real world application, in most cases, the lower level optimizer cannot be expressed in closed form.

3.2 Case 2 Toll Setting: Reformulation using optimality condition

The following example is adapted from (Li, 2024) and (Beck and Schmidt, 2021).

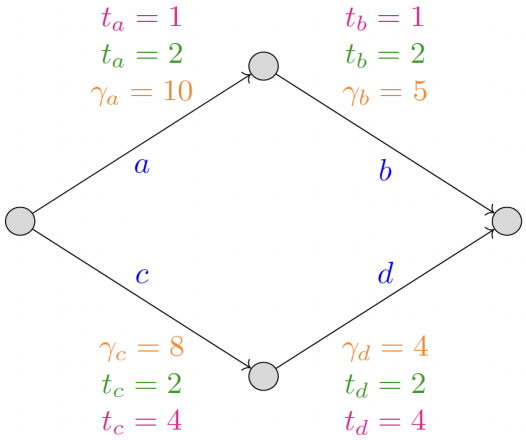


Figure 1: Toll Problem

Suppose a group of drivers want to travel from the leftmoset node to the right mode node and all of them want to minimize the cost. In this situation, costs can, e.g., be travel

time, toll costs, or a combination of both. On the other hand, there is a toll setting agency, which decides on the tolls imposed on certain parts of the highway system. This toll setting agency wants to maximize the revenues based on the tolls and the travelers assuming that all travelers act optimally to minimize their traveling costs. Abstractly, the toll setting agency can be viewed as the leader and the travelers are the followers.

Let γ_{ij} denote the base travel cost (excluding tolls) between node i and node j. Let D represent the demand or the number of travelers seeking to travel from the origin to the destination. Let E be the set of edges (road segments) on which tolls can be imposed. Let V denote the set of nodes in the network. Let P be the set of all paths from the origin to the destination. Furthermore, let x_p denote the number of travelers choosing path p and t_{ij} represent the toll set on edge (i, j).

Given the above information, the optimization problem is formulated as

$$\max_{\mathbf{t}} \sum_{(i,j)\in E} t_{ij} \cdot \sum_{p\in P:(i,j)\in p} x_p^* \quad \text{s.t. } t_{ij} \geq 0, \quad \forall (i,j) \in \mathbf{E} \tag{19}$$

$$x^* = \arg\min_{x} \left\{ \sum_{p\in P} x_p \cdot \left(\sum_{(i,j)\in p} (\gamma_{ij} + t_{ij}) \right) \right\} \quad \text{s.t.} \sum_{p\in P} x_p = D \ , \ x_p \geq 0 \quad \forall p \in P \tag{20}$$

Briefly, the upper level problem (19) seeks to maximize toll revenues considering the response of the travelers (lower level problem (20)), who minimize their total travel cost, which includes tolls.

In this setup, it's possible that the optimal solution set of lower level problem (20) will not be a singleton and the problem can be divided into two sub-cases:

- **Optimistic formulation:** When multiple responses are optimal, the follower takes the decision that work best (e.g the route with the highest toll) for the leader. The reformulated problem is

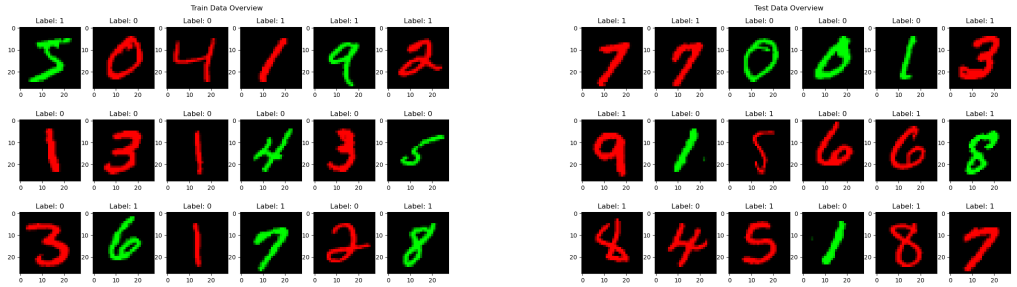
$$\max_{\mathbf{t}, \mathbf{x}^* \in \mathcal{S}(\mathbf{x})} \sum_{(i,j)\in E} t_{ij} \cdot \sum_{p\in P:(i,j)\in p} x_p^* \quad \text{s.t. } t_{ij} \geq 0, \quad \forall (i,j) \in \mathbf{E} \tag{21}$$

  where $\mathcal{S}(\mathbf{x})$ denotes the set of all optimal solution for the lower level problem (20).

- **Pessimistic formulation:** When multiple responses are optimal, the follower takes the decision that work worst (e.g the route with the lowest toll) for the leader. The reformulated problem is

$$\max_{\mathbf{t}} \min_{\mathbf{x}^* \in \mathcal{S}(\mathbf{x})} \sum_{(i,j)\in E} t_{ij} \cdot \sum_{p\in P:(i,j)\in p} x_p^* \quad \text{s.t. } t_{ij} \geq 0, \quad \forall (i,j) \in \mathbf{E} \tag{22}$$

Since the optimistic formulation is usually easier to solve than the pessimistic formulation, it's commonly used. To represent the optimal solution $\mathcal{S}(\mathbf{x})$ for the lower problem (which is the same for both optimistic and pessimistic formulation), the optimality condition of the lower problem is used as explicit constraints for the upper problem.



(a) Train Data (b) Test Data

Figure 2: Dataset Overview

3.3 Case 3 Invariant Representation Learning

The following example is adapted from (Zhang et al., 2023a).

Bilevel optimization can be used to improve the domain generalizability of machine learning model. Training model using Invariant risk minimization(IRM) (Arjovsky et al., 2020) can improve model robustness to distribution shifts. Specifically, the upper-level task of IRM is representation learning, which tries to find domain-independent representation and the lower-level task is to train a prediction head for classification or regression based on the learned representation.

Mathematically, the optimization goal can be formulated as the following format shown in (Zhang et al., 2023a).

$$\min_{\theta} \sum_{i=1}^{E} \ell_i(\phi^*(\theta) \circ \theta) \text{ s.t. } \phi^*(\theta) \in \arg\min_{\phi} \ell_i(\phi \circ \theta) \tag{23}$$

where $\phi \circ \theta$ denotes the whole model, with ϕ being prediction head and θ being representation learning backbone. l_i is loss function for domain i and E is the total number of training domain.

Due to the difficulty of optimizing (23) directly, (23) is usually relaxed to a single-level optimization known as IBMv1. The relaxed format is

$$\min_{\theta} \sum_{i=1}^{E} [\ell_i(\theta) + \gamma \|\nabla_{w|w=1.0} \ell_i(w \circ \theta)\|_2^2] \tag{24}$$

where γ is the regularization strength and the gradient is evaluated at $W = 1$ (identity prediction head). Specifically, instead of training a predictor as lower level task, the model is asked to generate a good embedding that should work with "identity" classification head (restricted to linear mapping only).

To show IRM can indeed improve the robustness of deep learning model, we constructed a controlled experiment using the MNIST dataset (LeCun et al., 2010) following the setup in (Arjovsky et al., 2020)

- **Dataset:** The MNIST dataset was randomly partitioned into three equal-sized environments: two for training and one for testing. Each image underwent a systematic transformation process:

  1. We binarized the labels by assigning y = 0 to digits 0-4 and y = 1 to digits 5-9. These labels were subject to random noise, with a 25% probability of being flipped.
  2. Based on the binary label, images were colored either red or green, depending on which environment they were in.
  3. The correlation between colors and labels varied across environments through an environment-specific color-flipping probability e: 20% for the first training environment, 10% for the second training environment, and 90% for the test environment

  This design created a challenging scenario where the model must learn to rely on genuine features rather than spurious color correlations to assign prediction (i.e. y = 0 for digits 0-4 and y = 1 for digits 5-9). An overview of the train and test data is shown in Figure2.

- **Model:** Two models with the same structure are trained for this experiment using empirical risk minimization and IRM respectively. The neural network model consists of four layers, with the input being processed with two two convolutional layers: the first with 3 input channels and 20 output channels, followed by a second layer with 20 input and 50 output channels. 5×5 kernels with unit stride was used in both layers. The extracted features are then flattened and passed through two fully connected layers: the first reduces the dimension to 500 units, and the final layer outputs a single scalar prediction.

- **Loss Function:** The first model is trained using the standard Empirical Risk Minimization and is named ERM for nomenclature simplicity. The other model is trained using IRM loss and is named IRM.

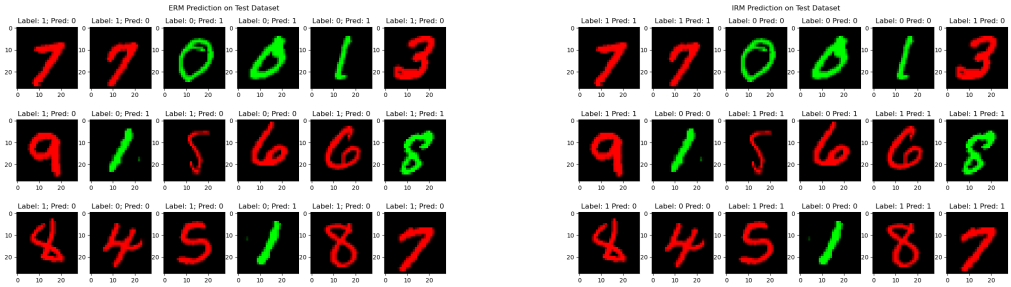
Model	Train Set	Test Set
ERM	85.14%	10.11%
IRM	72.56%	61.60%

Table 1: Performance comparison between ERM and IRM

The experiment results can be found in Table1 and Figure 3. Apparently, using IRM loss helped model learn an invariant representation across environment and greatly improved model robustness and generalizability. The code used for reproducing the above results are adapted from (Kano, 2019).

4 Conclusion

The field of bilevel optimization is still evolving rapidly. Recent advances suggest exciting applications in areas such as neural architecture search (Tu et al., 2023), meta-learning



(a) ERM Prediction (b) IRM Prediction

Figure 3: Test Result Overview

(Rajeswaran et al., 2019), and robust model training. This short tutorial gives a gentle introduction on both theoretical foundations and practical application, hoping to provide an accessible entry point for researchers who are not familiar with bilevel optimization. As the community continues to address computational challenges and theoretical gaps, we anticipate bilevel optimization will play an increasingly crucial role in advancing machine learning capabilities. We hope this tutorial serves as a stepping stone for readers to engage with and contribute to this dynamic field.

5 Acknowledgments

The report is polished using AI tools (e.g., GPT) for better style and language. The content, both theory and application, and writing are all done by the author independently.

References

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization, 2020. URL https://arxiv.org/abs/1907.02893.

Yasmine Beck and Martin Schmidt. A gentle and incomplete introduction to bilevel optimization. Technical report, Trier University, 2021.

Caroline Crockett and Jeffrey A. Fessler. Bilevel methods for image reconstruction. *Foundations and Trends® in Signal Processing*, 15(2–3):121–289, 2022. ISSN 1932-8354. doi: 10.1561/2000000111. URL http://dx.doi.org/10.1561/2000000111.

Reiichiro Kano. invariant-risk-minimization. https://github.com/reiinakano/invariant-risk-minimization, 2019.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Can Li. Lecture 24: Bilevel optimization. Lecture Notes in ChE 597: Computational Optimization, 2024.

Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. Meta-learning with implicit gradients, 2019. URL https://arxiv.org/abs/1909.04630.

George B. Richardson. The theory of the market economy. *Revue économique*, 46(6):1487–1496, 1995. ISSN 00352764, 19506694. URL http://www.jstor.org/stable/3502458.

Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots. Truncated back-propagation for bilevel optimization, 2019. URL https://arxiv.org/abs/1810.10667.

Sidak Pal Singh and Dan Alistarh. Woodfisher: Efficient second-order approximation for neural network compression, 2020. URL https://arxiv.org/abs/2004.14340.

Chongjun Tu, Peng Ye, Weihao Lin, Hancheng Ye, Chong Yu, Tao Chen, Baopu Li, and Wanli Ouyang. Efficient architecture search via bi-level data pruning, 2023. URL https://arxiv.org/abs/2312.14200.

Yihua Zhang, Prashant Khanduri, Ioannis Tsaknakis, Yuguang Yao, Mingyi Hong, and Sijia Liu. An introduction to bi-level optimization: Foundations and applications in signal processing and machine learning. *arXiv preprint arXiv:2308.00788*, 2023a.

Yihua Zhang, Yuguang Yao, Parikshit Ram, Pu Zhao, Tianlong Chen, Mingyi Hong, Yanzhi Wang, and Sijia Liu. Advancing model pruning via bi-level optimization, 2023b. URL https://arxiv.org/abs/2210.04092.

Simiao Zuo, Chen Liang, Haoming Jiang, Xiaodong Liu, Pengcheng He, Jianfeng Gao, Weizhu Chen, and Tuo Zhao. Adversarial regularization as stackelberg game: An unrolled optimization approach. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6562–6577, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.527. URL https://aclanthology.org/2021.emnlp-main.527.