Bilevel Optimization for Hyperparameter Learning in Supporting Vector Machines

Lei Huang
Department of Mathematics
University of California, San Diego
La Jolla, CA 92093
`leh010@ucsd.edu`

Jiawang Nie
Department of Mathematics
University of California, San Diego
La Jolla, CA 92093
`njw@math.ucsd.edu`

Jiajia Wang
Department of Mathematics
University of California, San Diego
La Jolla, CA 92093
`jiw133@ucsd.edu`

Suhan Zhong
School of Mathematical Sciences
Shanghai Jiao Tong University
Shanghai, China, 200240
`suzhong@sjtu.edu.cn`

Abstract

Bilevel optimization is central to many machine learning tasks, including hyperparameter learning and adversarial training. We present a novel single-level reformulation for bilevel problems with convex lower-level objective functions and linear constraints. Our method eliminates auxiliary Lagrange multiplier variables by expressing them in terms of the original decision variables, which allows the reformulated problem to preserve the same dimension as the original problem. We applied our method to support vector machines (SVMs) and evaluated it on several benchmark tasks, demonstrating efficiency and scalability.

1 Introduction

Bilevel optimization models a hierarchical decision-making process involving a *leader* and a *follower*. The leader's decision depends on the follower's optimal response, which is determined by solving the follower's problem parametrized by the leader's decision. This structure is prevalent in many machine learning tasks. For example, in hyperparameter learning of support vector machines (SVMs), which are supervised learning methods that construct an optimal separating hyperplane to maximize the margin between different classes, the leader tunes hyperparameters while the follower trains model parameters [1, 2]. SVMs typically involve choosing regularization parameters and kernel parameters that crucially affect the separating hyperplane, and this selection naturally leads to a bilevel optimization formulation: the upper-level objective minimizes a validation loss over hyperparameters, while the lower-level problem is the convex quadratic program that fits the classifier (finding w, b, ξ given the hyperparameters) [3]. In adversarial training, the upper-level updates model weights while the lower-level generates adversarial perturbations [4]. We also refer to [5–10] for applications in economics, energy markets and machine learning. In practice, the lower-level problem is typically convex and subject to linear constraints.

In this paper, we consider the bilevel optimization

$$\min_{(x,y)\in X} \; F(x,y) \quad s.t. \quad y \in S(x), \tag{1.1}$$

where $S(x)$ is the optimizer set of the *lower-level problem*

$$(P_x): \quad \min_{y} \; f(x,y) \quad s.t. \quad Ay - b(x) \geq 0.$$

In the above, $A \in \mathbb{R}^{m\times p}$, $x \in \mathbb{R}^n$ is the upper-level variable, $y \in \mathbb{R}^p$ is the lower-level variable, $b(x) = (b_1(x), \dots, b_m(x))^\top$ is a vector-valued function and $X \subseteq \mathbb{R}^n \times \mathbb{R}^p$ is the constraint set. Throughout this paper, we assume that $F, f, b_1, \dots, b_m$ are continuously differentiable.

Many numerical algorithms have been proposed to solve (1.1) such as descent algorithms [11–13], penalty and trust-region algorithms [14–16] and semidefinite relaxations [17]. One important policy is to reformulate bilevel optimization into mathematical program with equilibrium constraints (MPEC) by replacing $y \in S(x)$ with the Karush-Kuhn-Tucker (KKT) conditions of (P_x) [18–21]. Then, optimization algorithms are applied to solve the related MPEC. Since the optimizer set $S(x)$ usually does not have an explicit expression, the MPEC reformulation typically introduces extra multiplier variables, which increases computational complexity.

Since the lower-level constraints are linear in y, every feasible point of (1.1) must satisfy the KKT conditions

$$\exists \lambda \in \mathbb{R}^m \quad s.t. \quad \nabla_y f(x,y) - A^\top \lambda = 0,\ 0 \le [Ay - b(x)] \perp \lambda \ge 0, \tag{1.2}$$

where $\lambda = (\lambda_1, \dots, \lambda_m)^\top$ denotes the Lagrange multipliers. A pair $(x, y) \in X$ satisfying (1.2) is called a KKT point of (1.1), and the corresponding (x, y, λ) is called a critical pair. The classical MPEC reformulation replaces $y \in S(x)$ by (1.2), i.e.,

$$\text{(MPEC)} \left\{ \begin{array}{ll} \min\limits_{(x,y,\lambda)} & F(x,y) \\ s.t. & (x,y) \in X,\ \lambda \in \mathbb{R}^m, \\ & \nabla_y f(x,y) - A^\top \lambda = 0, \\ & 0 \le [Ay - b(x)] \perp \lambda \ge 0, \end{array} \right. \tag{1.3}$$

which introduces λ as new variables. In this work, we exploit how to represent λ as explicit functions of (x, y). This trick is called the *Lagrange Multiplier Expressions (LMEs)*, and has been widely applied to sovle different optimization problems [17, 22–27].

Our major contributions are summarized as follows.

- We present a novel LME-MPEC reformulation to solve linearly constrained bilevel optimization with Lagrange multiplier expressions.
- We propose a penalty algorithm to solve this LME-MPEC reformulation and prove its convergence properies.
- We study a modified support vector machine (SVM) model within the bilevel framework and testify the efficiency of our method on several benchmark tasks.

Notation

For a matrix A, $A^\top$ denotes its transpose, $\mathrm{rank}(A)$ its rank, and $\mathrm{Nul}(A)$ its null space. If A is invertible, A^{-1} denotes its inverse. For integers m, n, let $\mathbf{0}_{m\times n}$ be the $m \times n$ zero matrix, $\mathbf{0}_m$ the m-dimensional zero vector, and I_n the $n \times n$ identity matrix. For a vector $v = (v_1, \dots, v_m)$, we write v_i for its i-th component, and $\mathrm{diag}(v) = \mathrm{diag}(v_1, \dots, v_m)$ for the diagonal matrix with v on its diagonal. The Euclidean norm of v is denoted by $\|v\|$. For a continuously differentiable function $f(x, y)$, we use ∇f to denote its full gradient and $\nabla_y f, \nabla_{y_i} f$ for its partial gradients in y, y_i, respectively.

2 Expressions of KKT Sets

Let $\mathcal{K}$ denote the set of KKT points that satisfies (1.2).

Definition 2.1. *If there exists a tuple of functions $\tau = (\tau_1, \dots, \tau_m)$ with each $\tau_i : X \to \mathbb{R}$ such that $\lambda = \tau(x, y)$ for every $(x, y) \in \mathcal{K}$, then τ is called a Lagrange multiplier expression (LME).*

LMEs almost always exist for the lower-level problem (P_x). Define

$$H(x,y) := \begin{bmatrix} A^\top \\ \mathrm{diag}(Ay - b(x)) \end{bmatrix}, \quad \hat{f}(x,y) := \begin{bmatrix} \nabla_y f(x,y) \\ \mathbf{0}_{m\times 1} \end{bmatrix}. \tag{2.1}$$

The KKT system (1.2) implies $H(x,y)\lambda = \hat{f}(x,y)$. If there exists a matrix function $L(x,y)$ such that $L(x,y)H(x,y) = I_m$, then we can obtain the LME

$$\lambda = \tau(x,y) = L(x,y)\hat{f}(x,y). \tag{2.2}$$

When $H(x,y)$ has full column rank, which is the general case, we can choose $L = H^{\dagger} := (H^\top H)^{-1}H^\top$, leading to

$$\lambda = \tau(x,y) = H(x,y)^{\dagger}\hat{f}(x,y). \tag{2.3}$$

If $\tau(x,y)$ is a LME of (1.1), then

$$\mathcal{K} = \big\{(x,y) \in X \;\big|\; \nabla_y f(x,y) - A^\top \tau(x,y) = 0,\;\; 0 \le [Ay - b(x)] \perp \tau(x,y) \ge 0\big\}.$$

We consider the following MPEC reformulation:

$$\text{(LME-MPEC)} \left\{ \begin{array}{ll} \min\limits_{(x,y)\in X} & F(x,y) \\ s.t. & \nabla_y f(x,y) - A^\top \tau(x,y) = 0, \\ & 0 \le [Ay - b(x)] \perp \tau(x,y) \ge 0. \end{array} \right. \tag{2.4}$$

This reformulation eliminates the Lagrange multiplier variables by exploiting LMEs, making it more computationally efficient than the classical MPEC reformulation, particularly when the LMEs can be computed easily. The below is an illustrating example.

Example 2.2. Consider the bilevel optimization problem

$$\left\{ \begin{array}{ll} \min & \sum\limits_{i=1}^{3} x_i^2 y_i^2 + 2(x_1^2 y_2^2 + x_2^2 y_3^2 + x_3^2 y_1^2) - 2(x_1x_2y_1y_2 + x_1x_3y_1y_3 + x_2x_3y_2y_3) \\ s.t. & x \in \mathbb{R}^3,\; y \in \mathbb{R}^3,\; y \in S(x), \end{array} \right. \tag{2.5}$$

where $S(x)$ is the solution set of the lower-level problem:

$$\left\{ \begin{array}{ll} \min\limits_{y\in\mathbb{R}^3} & -x^\top x + y^\top y \\ s.t. & \sum\limits_{i=1}^{3} x_i - \sum\limits_{i=1}^{3} y_i \ge 0. \end{array} \right.$$

Since the lower-level problem is convex, the global optimal value can be obtained by solving its MPEC reformulations. For

$$H(x,y) = \begin{bmatrix} -1 \\ -1 \\ -1 \\ \sum\limits_{i=1}^{3} x_i - \sum\limits_{i=1}^{3} y_i \end{bmatrix}, \; \hat{f}(x,y) = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ 0 \end{bmatrix}, \quad L(x,y) = \begin{bmatrix} -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & 0 \end{bmatrix},$$

it is easy to verify $H(x,y)L(x,y) = 1$, thus the LME $\lambda = L(x,y)\hat{f}(x,y)$. The MPEC reformulation (1.3) and LME-MPEC reformulation (2.4) of this problem are both polynomial optimization problems, which can be solved globally by Moment-Sum-of-Squares relaxations. We implemented the computation using the software [28], which calls the SDP package SeDuMi [29]. The minimum value of (2.5) is $F_{\min} = 0$, achieved at $x = (0.1125, 0.1125, 0.1125)^\top$, $y = (-0.0014, -0.0014, -0.0014)^\top$. It only took 0.1752 second to solve the LME-MPEC reformulation, while solving the standard MPEC formulation takes approximately 1.4511 second.

For some common constraint sets such as box and simplex constraints, convenient expressions for Lagrange multipliers can be derived. The construction details can be found in [23, Chapter 6.2].

Simple constraints. When $g(y) = Ay - b$ with $b \in \mathbb{R}^m$, $H(y) = \begin{bmatrix} A^\top \\ \operatorname{diag}(Ay - b) \end{bmatrix}$. If $H(y)$ has full column rank for all $y \in \mathbb{C}^p$, then there exists a polynomial matrix $L(y)$ of degree at most $m - \operatorname{rank}(A)$ such that $L(y)H(y) = I_m$. Such an $L(x)$ can be solved via linear programming.

Equality constraints. When $g(x,y) = \begin{bmatrix} Ay - b(x) \\ -Ay + b(x) \end{bmatrix}$, (1.2) can be simplified to $\nabla_y f(x,y) = A^\top \lambda$. In this case, λ can be solved by symbolic Gaussian elimination. For the special case that $A^\top$ has full column rank, we have $\tau(x,y) = (AA^\top)^{-1} A \nabla_y f(x,y)$.

Box constraints. Let $l = (l_1, \ldots, l_p)$, $u = (u_1, \ldots, u_p)$ be distinct function tuples. If $A = \begin{bmatrix} I_p \\ -I_p \end{bmatrix}$, $b(x) = \begin{bmatrix} l(x) \\ -u(x) \end{bmatrix}$, then we have the LME

$$\tau_i(x,y) = \begin{cases} \dfrac{(u_i(x) - y_i)\nabla_{y_i} f(x,y)}{u_i(x) - l_i(x)}, & i = 1, \ldots, p, \\ \dfrac{(l_{i-p}(x) - y_{i-p})\nabla_{y_{i-p}} f(x,y)}{u_{i-p}(x) - l_{i-p}(x)}, & i = p+1, \ldots, 2p. \end{cases}$$

Simplex constraints. Let $l = (l_1, \ldots, l_p)$ be a function tuple and u_0 a scalar function. If $A = \begin{bmatrix} I_p \\ -e^\top \end{bmatrix}$, $b(x) = \begin{bmatrix} l(x) \\ -u_0(x) \end{bmatrix}$, then we have the LME

$$\tau_i(x,y) = \begin{cases} \dfrac{(u_0(x) - y_i)\nabla_{y_i} f(x,y)}{u_0(x) - l_1(x) - \cdots - l_p(x)}, & i = 1, \ldots, p, \\ \dfrac{(l(x) - y)^\top \nabla_y f(x,y)}{u_0(x) - l_1(x) - \cdots - l_p(x)}, & i = p+1. \end{cases}$$

3 A Penalty Algorithm for solving (2.4)

In this section, we give a penalty method to solve the LME-MPEC (2.4). It is known that classical constraint qualifications (CQs) such as linear independence constraint qualification (LICQ) and Mangasarian–Fromovitz constraint qualification (MFCQ) typically fail for MPECs [30]. In the following, we define the MFCQ-type CQ similarly as in [31]. For simplicity, we assume X is the whole space, i.e., $X = \mathbb{R}^n \times \mathbb{R}^p$.

Let $\tau = (\tau_1, \ldots, \tau_m)$ be a LME of (1.2). Denote $g(x,y) = Ay - b(x)$ with $g_i(x,y) = (Ay - b(x))_i$ for all $i \in [m]$ and

$$\phi := \nabla_y f - A^\top \tau \quad \text{with} \quad \phi_j(x,y) := \nabla_{y_j} f(x,y) - \sum_{i=1}^m \tau_i(x,y) \nabla_{y_j} g_i(x,y) \ (j \in [p]). \tag{3.1}$$

For each $(x^*, y^*) \in X$, define the index sets

$$\begin{aligned} \alpha(x^*,y^*) &:= \{i \in [m] : g_i(x^*,y^*) = 0,\ \tau_i(x^*,y^*) > 0\}, \\ \gamma(x^*,y^*) &:= \{i \in [m] : g_i(x^*,y^*) > 0,\ \tau_i(x^*,y^*) = 0\}, \\ \beta(x^*,y^*) &:= \{i \in [m] : g_i(x^*,y^*) = 0,\ \tau_i(x^*,y^*) = 0\}. \end{aligned}$$

A relaxed problem of (2.4) can be given as

$$\text{(LME-MPEC-R)} \begin{cases} \min\limits_{(x,y)} & F(x,y) \\ s.t. & \phi_j(x,y) = 0, & j \in [p], \\ & g_i(x,y) = 0,\ \tau_i(x,y) \ge 0, & i \in \alpha(x^*,y^*), \\ & g_i(x,y) \ge 0,\ \tau_i(x,y) = 0, & i \in \gamma(x^*,y^*), \\ & g_i(x,y) \ge 0,\ \tau_i(x,y) \ge 0, & i \in \beta(x^*,y^*). \end{cases} \tag{3.2}$$

We then introduce a specific MFCQ-type CQ for (3.2).

Definition 3.1. *The MFCQ-LME-R condition is said to be satisfied at (x^*,y^*) if the set of vectors*

$$\{\nabla\phi_j(x^*,y^*) : j \in [p]\} \cup \{\nabla g_i(x^*,y^*) : i \in \alpha(x^*,y^*)\} \cup \{\nabla\tau_i(x^*,y^*) : i \in \gamma(x^*,y^*)\}$$

is linearly independent, and there exists a vector $d \in \mathbb{R}^{n+p}$ such that

$$\begin{cases} \nabla\phi_j(x^*,y^*)^\top d = 0, & j \in [p], \\ \nabla g_i(x^*,y^*)^\top d = 0, & i \in \alpha(x^*,y^*), \\ \nabla\tau_i(x^*,y^*)^\top d = 0, & i \in \gamma(x^*,y^*), \\ \nabla g_i(x^*,y^*)^\top d > 0,\ \nabla\tau_i(x^*,y^*)^\top d > 0 & i \in \beta(x^*,y^*). \end{cases} \tag{3.3}$$

In this work, we define *Strongly (S)-stationary* points.

Definition 3.2. *A feasible point (x^*, y^*) of the LME-MPEC (2.4) is said to be strongly (S)-stationary if there exist multipliers (λ, μ, ν) with $\lambda \in \mathbb{R}^p$, $\mu = (\mu_1, \ldots, \mu_m) \in \mathbb{R}^m$ and $\nu = (\nu_1, \ldots, \nu_m) \in \mathbb{R}^m$ such that*

$$\left\{\begin{array}{l} \nabla F(x^*, y^*) = \nabla \phi(x^*, y^*)^\top \lambda + \nabla g(x^*, y^*)^\top \mu + \nabla \tau(x^*, y^*)^\top \nu, \\ \mu_i = 0 \ \forall i \in \gamma(x^*, y^*), \quad \nu_j = 0 \ \forall j \in \alpha(x^*, y^*), \\ \mu_k \geq 0, \, \nu_k \geq 0, \, \forall k \in \beta(x^*, y^*). \end{array}\right. \tag{3.4}$$

By penalizing complementarity constraints, we can transform (2.4) into the following standard nonlinear program:

$$\text{(LME-MPEC-P)} \left\{\begin{array}{ll} \min\limits_{(x,y)} & F(x,y) + \theta \cdot g(x,y)^\top \tau(x,y) \\ s.t. & \phi(x,y) = 0, \, g(x,y) \geq 0, \, \tau(x,y) \geq 0, \end{array}\right. \tag{3.5}$$

where $\theta > 0$ is a penalty parameter. It is easy to see that the MFCQ-LME-R at an S-stationary point of (3.2) implies the MFCQ of (3.5).

Theorem 3.3. *For a given $\theta > 0$, if the MFCQ-LME-R holds at an S-stationary point (x^*, y^*) of (3.2), then the MFCQ holds at (x^*, y^*) of* (3.5).

We introduce the concept of ε-KKT points of (3.5).

Definition 3.4. *A feasible point (x, y) of* (3.5) *is called an ε-KKT point if there exist multipliers (λ, μ, ν) with $\lambda \in \mathbb{R}^p$, $\mu \in \mathbb{R}^m$, $\nu \in \mathbb{R}^m$ such that*

$$\begin{array}{l} \|\nabla F(x) + \theta \nabla (\tau(x,y)^\top g(x,y)) - \nabla \phi(x,y)^\top \lambda - \nabla g(x,y)^\top \mu - \nabla \tau(x,y)^\top \nu\| \leq \varepsilon, \\ 0 \leq \mu \perp g(x,y) \geq 0, \quad 0 \leq \nu \perp \tau(x,y) \geq 0. \end{array}$$

We then propose a penalty algorithm for solving bilevel optimization.

Algorithm 1 An iterative penalty method for solving LME-MPEC

```
Require: a starting point $(x^{(0)}, y^{(0)})$, an initial penalty parameter $\theta_1 > 0$, an initial KKT tolerance
  $\varepsilon_1 > 0$ and an exit tolerance $\varepsilon^* > 0$. Set $t = 0$.
  while $\tau(x^{(t)}, y^{(t)})^\top g(x^{(t)}, y^{(t)}) \leq \varepsilon^*$ do
      Searching from $(x^{(t)}, y^{(t)})$, find an $\varepsilon_{t+1}$-KKT point $(x^{(t+1)}, y^{(t+1)})$ of (3.5) with $\theta := \theta_{t+1}$.
      Update $t := t + 1$. Set $\theta_{t+1} > \theta_t$ and $\varepsilon_{t+1} < \varepsilon_t$.
  end while
  Output the candidate solution $(x^{(t)}, y^{(t)})$.
```

In practice, we initialize the parameters as $\theta_1 = \varepsilon_1 = 0.1, \varepsilon^* = 0.01$, and update them according to

$$\varepsilon_{t+1} = r_1 \varepsilon_t, \qquad \theta_{t+1} = r_2 \theta_t,$$

where $0 < r_1 < 1 < r_2$. Below, we analyze the convergence of Algorithm 1. The proofs are deferred to Appendix.

Theorem 3.5. *Assume F, f, b, τ are continuously differentiable. Let $\{(x^{(t)}, y^{(t)})\}_{t=1}^\infty$ be the sequence generated by Algorithm 1. Assume that $\lim_{t \to \infty} (x^{(t)}, y^{(t)}) = (x^*, y^*)$.*

(i) Suppose $\varepsilon_t \to 0$, $0 \leq \tau(x^*, y^*) \perp g(x^*, y^*) \geq 0$ and the MFCQ-LME-R condition holds at (x^*, y^*). Then (x^*, y^*) is a S-stationary point of (2.4).

(ii) Suppose F is bounded from below and attains its minimum. If each $(x^{(t)}, y^{(t)})$ is a global solution of (3.5) *with $\theta = \theta_t$, then (x^*, y^*) satisfies the complementarity condition $\tau_i(x^*, y^*) g_i(x^*, y^*) = 0$ for all $i \in [m]$.*

4 Application: Hyperparameter Optimization

Support vector machine (SVM) is a supervised learning method that constructs an optimal separating hyperplane to maximize the margin between different classes. The hyperparameter optimization [32] of SVM can be formulated as a bilevel optimization problem. Let $\mathcal{D}_{val} = \{(x^i_{val}, y^i_{val})\}_{i=1}^{N_{val}}$ and $\mathcal{D}_{tr} = \{(x^i_{tr}, y^i_{tr})\}_{i=1}^{N_{tr}}$ denote the validation and training datasets, respectively. In these two datasets, each $x^i_{val}, x^i_{tr} \in \mathbb{R}^p$, $y^i_{val}, y^i_{tr} \in \mathbb{R}$ and N_{val}, N_{tr} denote the numbers of samples in the validation and training datasets respectively. The hyperparameter optimization of SVM takes the form of

$$\min_{(c,w,b,\xi)} \frac{1}{N_{val}} \sum_{i=1}^{N_{val}} \sigma(-(y^i_{val}(w^\top x^i_{val}) + b)/\|w\|^2) \quad s.t. \quad (w, b, \xi) \in S(c),$$

where $\sigma(\cdot)$ stands for the sigmoid function and $S(c)$ is the solution set of the SVM problem

$$\begin{cases} \min\limits_{(w,b,\xi)} & f(c, w, b, \xi) = \frac{1}{2}\|w\|^2 + \frac{1}{2}\sum\limits_{i=1}^{N_{tr}} e^{c_i}\xi_i^2 \\ s.t. & y^i_{tr}(w^\top x^i_{tr} + b) \geq 1 - \xi_i, \quad \forall i \in [N_{tr}]. \end{cases}$$

In the above, $c = (c_1, \ldots, c_{N_{tr}})^\top \in \mathbb{R}^{N_{tr}}$ is the vector of hyperparameters, and $w \in \mathbb{R}^p$, $b \in \mathbb{R}$, $\xi \in \mathbb{R}^{N_{tr}}$ are parameters of the training classifier. The term $-(y^i_{val}(w^\top x^i_{val}) + b)/\|w\|^2$ represents the negative signed distance from the point (x^i_{val}, y^i_{val}) to the hyperplane $\{\, x \in \mathbb{R}^p \mid w^\top x + b = 0 \,\}$. It is clear that the lower-level SVM problem is linearly constrained, where the coefficient matrix is

$$A = \begin{bmatrix} y^1_{tr}(x^1_{tr})^\top & y^1_{tr} & 1 & 0 & \cdots & 0 \\ y^2_{tr}(x^2_{tr})^\top & y^2_{tr} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ y^{N_{tr}}_{tr}(x^{N_{tr}}_{tr})^\top & y^{N_{tr}}_{tr} & 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{4.1}$$

It is clear that A has full row rank, thus we can set

$$\tau(w, b, \xi) = \begin{bmatrix} \mathbf{0}_{(p+1)\times(N_{tr})} \\ I_{N_{tr}} \end{bmatrix} \nabla_{w,b,\xi} f(c, w, b, \xi).$$

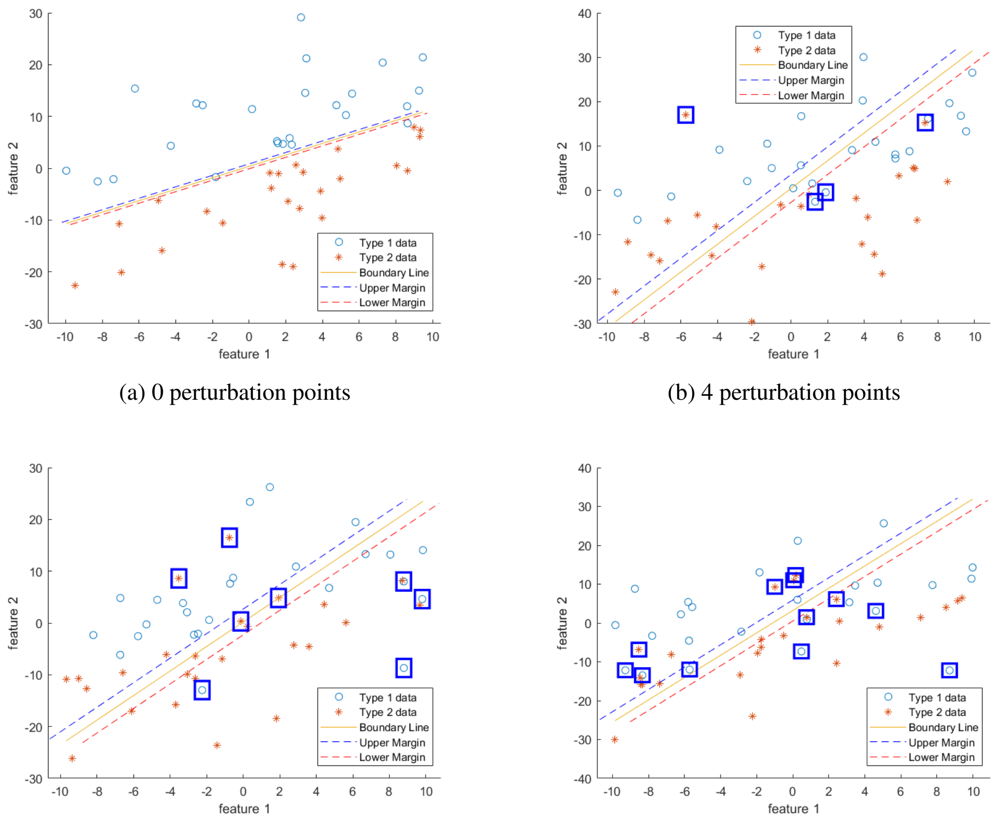
We test Algorithm 1 by performing a series of classification tasks on both low-dimensional datasets (for visualization) and high-dimensional datasets (to evaluate scalability). All experiments are conducted using MATLAB R2024a on a desktop equipped with Intel Core i5-14600K CPU (3.50 GHz) and 32 GB RAM. In each iteration, the subproblem (3.5) is solved using the `fmincon` function from MATLAB Optimization Toolbox [33]. We initialize the parameters as $\theta_1 = \varepsilon_1 = 0.1$ and set the stopping threshold to $\varepsilon^* = 0.01$. We also set $r_1 = 0.5$, $r_2 = 2$. We set the initial hyperparameter $c^{(0)}$ to be the zero vector. The initial parameters $(w^{(0)}, b^{(0)}, \xi^{(0)})$ are chosen by solving the following quadratic optimization problem:

$$\min \|w\|^2 + \|b\|^2 + \|\xi\|^2 \quad s.t. \quad A \begin{bmatrix} w \\ b \\ \xi \end{bmatrix} = e, \tag{4.2}$$

where $e \in \mathbb{R}^{N_{tr}}$ is the vector of all ones. This ensures that the algorithm starts from a point in the feasible region of the lower-level problem.

Synthetic Data with Perturbations. We generated 50 nearly linearly separable two-dimensional data points using `randn` in MATLAB. To test robustness, we trained linear SVM's decision boundary on datasets with label perturbations. For each test, the dataset was partitioned into a 60% training set, a 10% validation set, and a 30% test set. The classifier was trained using Algorithm 1. The resulting decision boundaries are presented in Figure 1, and the classification accuracy is summarized in Table 1. In Figure 1, clean data points from Type 1 and Type 2 are shown in blue circles and red stars, respectively. The perturbed points, where the original class label was flipped, are highlighted with a blue border. The decision boundary is plotted in a solid line and upper/lower margins are plotted in dashed lines. It shows that our method finds the perfect linear separation for unperturbed datasets, while preserving high accuracy for datasets with label perturbations.

Fisher's Dataset. Consider the Fisher Iris dataset [34], which contains 150 instances from three species of iris plants, each with four features. We restrict the problem to a two-class, two-feature classification task using the last two features and the *setosa* and *virginica* classes. In our experiments, the dataset was partitioned into a 60% training set, a 10% validation set and a 30% test set, and the classifier was trained using Algorithm 1. In Figure 2, we presented the resulting decision boundaries



(a) 0 perturbation points

(b) 4 perturbation points

(c) 8 perturbation points

(d) 12 perturbation points

Figure 1: Decision boundaries on synthetic data with different number of perturbation points.

Table 1: Classification accuracy and runtime on synthetic data.

# of Perturbation Points	Accuracy	Runtime (s)
0	1.0000	0.2641
4	0.9375	0.2812
8	0.8750	0.2341
12	0.8125	0.3032

in the projected space, where *setosa* data points are plotted in blue circles, *virginica* data points are plotted in red stars, the decision boundary is plotted in a solid line and upper/lower margins are plotted in dashed lines. The classification accuracy is 0.9677, and the training process takes around 0.5241 second in total.

High-Dimensional Datasets. To evaluate the scalability of our proposed method, we conduct experiments on three widely used high-dimensional benchmark datasets: Ionosphere [35], Spambase [36], and MNIST [37]. These datasets were chosen due to their diversity in feature dimensions, sizes, and application domains. Specifically, Ionosphere is derived from radar signal processing, Spambase is a classical text classification dataset for spam detection, and MNIST is an image classification dataset where we restrict attention to digits 1 and 2 for binary classification. To assess generalization, we evaluate our algorithm under various data splits. In each split, the dataset is randomly divided into training, validation, and testing sets according to specified ratios, see Table 2 for details. The classification accuracy of the resulting classifiers are also reported in Table 2. We can observe that the proposed method demonstrates strong scalability to high-dimensional problems, while maintaining competitive accuracy across a range of data splits and datasets.

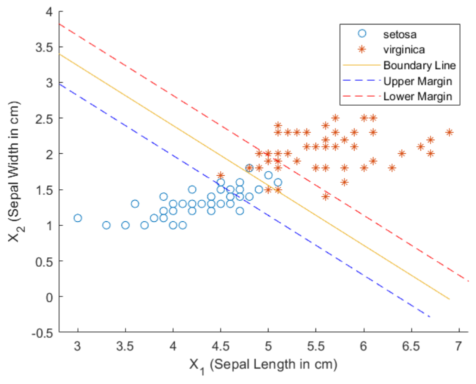


Figure 2: Decision boundaries on the Fisher Iris dataset using the last two features.

Table 2: Classification accuracy on high-dimensional datasets.

Split ratio	Ionosphere	Spambase	MNIST
0.8 : 0.1 : 0.1	0.9167	0.9020	0.9608
0.7 : 0.1 : 0.2	0.9014	0.9010	0.9608
0.6 : 0.1 : 0.3	0.8962	0.8874	0.9404
0.6 : 0.2 : 0.2	0.8873	0.8911	0.9505
0.5 : 0.2 : 0.3	0.8491	0.8808	0.9535

5 Conclusion

We present a novel approach to solve linearly constrained bilevel optimization. This method employs Lagrange multiplier expressions to express the lower-level optimality conditions, leading to a novel LME-MPEC reformulation. We propose a penalty algorithm to solve it and prove that the iterative sequence converges to an S-stationary point under mild conditions. The proposed method is applied to solve a hyperparameter support vector machine (SVM) model and its effectiveness is validated on multiple datasets.

Acknowledgments and Disclosure of Funding

Jiawang Nie and Jiajia Wang are partially supported by the NSF grant DMS-2110780.

References

[1] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1568–1577. PMLR, 2018.

[2] Stefano Coniglio, Anthony Dunn, Qingna Li, and Alain Zemkoho. Bilevel hyperparameter optimiza-tion for nonlinear support vector machines [j]. *Optimization Online: https://optimization-online. org*, pages 1–78, 2023.

[3] Qingna Li, Zhen Li, and Alain Zemkoho. Bilevel hyperparameter optimization for support vector classification: theoretical analysis and a solution method. *Mathematical Methods of Operations Research*, 96(3):315–350, 2022.

[4] Yihua Zhang, Guanhua Zhang, Prashant Khanduri, Mingyi Hong, Shiyu Chang, and Sijia Liu. Revisiting and advancing fast adversarial training through the lens of bi-level optimization. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 26693–26712. PMLR, 2022.

[5] Risheng Liu, Pan Mu, Xiaoming Yuan, Shangzhi Zeng, and Jin Zhang. A generic first-order algorithmic framework for bi-level programming beyond lower-level singleton. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6305–6315. PMLR, 2020.

[6] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1540–1552. PMLR, 2020.

[7] Matthew MacKay, Paul Vicol, Jon Lorraine, David Duvenaud, and Roger Grosse. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. *arXiv preprint arXiv:1903.03088*, 2019.

[8] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[9] Răzvan Andonie. Hyperparameter optimization in learning systems. *Journal of Membrane Computing*, 1(4):279–291, 2019.

[10] Monica and Parul Agrawal. A survey on hyperparameter optimization of machine learning models. In *2024 2nd International Conference on Disruptive Technologies (ICDT)*, pages 11–15. IEEE, 2024.

[11] Charles D Kolstad and Leon S Lasdon. Derivative evaluation and computational experience with large bilevel mathematical programs. *Journal of Optimization Theory and Applications*, 65 (3):485–499, 1990.

[12] Floriane Mefo Kue, Thorsten Raasch, and Alain B Zemkoho. A semismooth newton-type method for bilevel programs with linear lower level problem. *Pure and Applied Functional Analysis*, 8(5):1437–1463, 2023.

[13] Ioannis Tsaknakis, Prashant Khanduri, and Mingyi Hong. An implicit gradient-type method for linearly constrained bilevel problems. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5438–5442. IEEE, 2022.

[14] Yo Ishizuka and Eitaro Aiyoshi. Double penalty method for bilevel optimization problems. *Annals of Operations Research*, 34(1):73–88, 1992.

[15] Douglas J White and G Anandalingam. A penalty function approach for solving bi-level linear programs. *Journal of Global Optimization*, 3(4):397–419, 1993.

[16] Patrice Marcotte, Gilles Savard, and Daoli Zhu. A trust region algorithm for nonlinear bilevel programming. *Operations Research Letters*, 29(4):171–179, 2001.

[17] Jiawang Nie, Li Wang, Jane J Ye, and Suhan Zhong. A Lagrange multiplier expression method for bilevel polynomial optimization. *SIAM Journal on Optimization*, 31(3):2368–2395, 2021.

[18] Yan Jiang, Xuyong Li, Chongchao Huang, and Xianing Wu. Application of particle swarm optimization based on CHKS smoothing function for solving nonlinear bilevel programming problem. *Applied Mathematics and Computation*, 219(9):4332–4339, 2013.

[19] Youngdae Kim, Sven Leyffer, and Todd Munson. MPEC methods for bilevel optimization problems. In *Bilevel Optimization: Advances and Next Challenges*, pages 335–360. Springer, 2020.

[20] Christian Kanzow. Nonlinear complementarity as unconstrained optimization. *Journal of Optimization Theory and Applications*, 88(1):139–155, 1996.

[21] Andreas Fischer. New constrained optimization reformulation of complementarity problems. *Journal of Optimization Theory and Applications*, 97(1):105–117, 1998.

[22] Jiawang Nie. Tight relaxations for polynomial optimization and Lagrange multiplier expressions. *Mathematical Programming*, 178(1):1–37, 2019.

[23] Jiawang Nie. *Moment and Polynomial Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2023.

[24] Jiawang Nie, Jane J Ye, and Suhan Zhong. PLMEs and disjunctive decompositions for bilevel optimization. *arXiv preprint arXiv:2304.00695*, 2023.

[25] Jiawang Nie and Xindong Tang. Convex generalized Nash equilibrium problems and polynomial optimization. *Mathematical Programming*, 198(2):1485–1518, 2023.

[26] Jiawang Nie, Xindong Tang, and Suhan Zhong. Rational generalized Nash equilibrium problems. *SIAM Journal on Optimization*, 33(3):1587–1620, 2023.

[27] Lei Huang, Jiawang Nie, Jiajia Wang, and Lingling Xie. Lagrange multiplier expressions for matrix polynomial optimization and tight relaxations. *arXiv preprint arXiv:2506.12579*, 2025.

[28] Didier Henrion and Jean-Bernard Lasserre. Detecting global optimality and extracting solutions in gloptipoly. In *Positive Polynomials in Control*, pages 293–310. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[29] Jos F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1-4):625–653, 1999.

[30] Jand J Ye, Daoli Zhu, and Qiji Jim Zhu. Exact penalization and necessary optimality conditions for generalized bilevel programming problems. *SIAM Journal on Optimization*, 7(2):481–507, 1997.

[31] Samuel Ward, Alain Zemkoho, and Selin Ahipasaoglu. Mathematical programs with complementarity constraints and application to hyperparameter tuning for nonlinear support vector machines. *arXiv preprint arXiv:2504.13006*, 2025.

[32] Siyuan Xu and Minghui Zhu. Efficient gradient approximation method for constrained bilevel optimization. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence and 35th Conference on Innovative Applications of Artificial Intelligence and 13th Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023.

[33] The MathWorks Inc. Optimization Toolbox version: 9.4 (R2022b), 2022.

[34] R. A. Fisher. Iris. UCI Machine Learning Repository, 1936.

[35] Wing S. Hutton L. Sigillito, V. and K. Baker. Ionosphere. UCI Machine Learning Repository, 1989.

[36] Mark Hopkins, Erik Reeber, George Forman, and Jaap Suermondt. Spambase. UCI Machine Learning Repository, 1999.

[37] LeCun Yann. MNIST handwritten digit database. *ATT Labs.*, 2010.

Appendix: Proof of Theorem 3.5

(i). Let $(\lambda^{(t)}, \mu^{(t)}, \nu^{(t)})$ be the Lagrange multiplier associated with $(x^{(t)}, y^{(t)})$ as in Definition 3.4. For each $i \in [m]$, $t = 1, 2, \cdots$, denote $a_t := (\lambda^{(t)}, \hat{\mu}^{(t)}, \hat{\nu}^{(t)})$, where

$$\hat{\mu}_i^{(t)} := \mu_i^{(t)} - \theta_t \tau_i(x^{(t)}, y^{(t)}), \quad \hat{\nu}_i^{(t)} := \nu_i^{(t)} - \theta_t g_i(x^{(t)}, y^{(t)}).$$

Then, we have

$$\Big\|\nabla F(x^{(t)}, y^{(t)}) - \sum_{i=1}^{p} \lambda_i^{(t)} \nabla \phi_i(x^{(t)}, y^{(t)}) - \sum_{i=1}^{m} \hat{\mu}_i^{(t)} \nabla g_i(x^{(t)}, y^{(t)}) - \sum_{i=1}^{m} \hat{\nu}_i^{(t)} \nabla \tau_i(x^{(t)}, y^{(t)})\Big\| \le \varepsilon_t. \tag{A.1}$$

We claim that the sequence $\{a_t\}_{t=1}^{\infty}$ has a convergent subsequence. Otherwise, $\{a_t\}_{t=1}^{\infty}$ must be unbounded, e.g., $\|a_t\| \to \infty$ as $t \to \infty$. The sequence of normalized vectors, $\{a_t/\|a_t\|\}_{t=1}^{\infty}$, is bounded and therefore has an accumulation point, say $\tilde{a} := (\tilde{\lambda}, \tilde{\mu}, \tilde{\nu})$. This accumulation point cannot be the zero vector, as that would imply a subsequence of $\{a_t\}$ converges to zero, which contradicts with the assumption that $\{a_t\}_{t=1}^{\infty}$ is unbounded. In other words, $\tilde{\mu}_i$ or $\tilde{\nu}_i$ is nonzero for at least one index $i \in \beta(x^*, y^*)$. By dividing (A.1) with $\|a_t\|$, we get

$$\Big\|\frac{\nabla F(x^{(t)}, y^{(t)})}{\|a_t\|} - \sum_{i=1}^{p} \frac{\lambda_i^{(t)} \nabla \phi_i(x^{(t)}, y^{(t)})}{\|a_t\|} - \sum_{i=1}^{m} \frac{\hat{\mu}_i^{(t)} \nabla g_i(x^{(t)}, y^{(t)}) + \hat{\nu}_i^{(t)} \nabla \tau_i(x^{(t)}, y^{(t)})}{\|a_t\|}\Big\| \leq \frac{\varepsilon_t}{\|a_t\|}.$$

Since F, f, b, τ are continuously differentiable and $\lim_{t\to\infty}(x^{(t)}, y^{(t)}) = (x^*, y^*)$, the above implies

$$\sum_{i=1}^{p} \tilde{\lambda}_i \nabla \phi_i(x^*, y^*) + \sum_{i\in\alpha(x^*,y^*)\cup\beta(x^*,y^*)} \tilde{\mu}_i \nabla g_i(x^*, y^*) + \sum_{i\in\gamma(x^*,y^*)\cup\beta(x^*,y^*)} \tilde{\nu}_i \nabla \tau_i(x^*, y^*) = 0. \tag{A.2}$$

Since the MFCQ-LME-R condition holds at (x^*, y^*), there exists a nonzero vector $d \in \mathbb{R}^{n+p}$ that satisfies (3.3). Apply the inner product with d to both sides of (A.2). We obtain

$$\sum_{i\in\beta(x^*,y^*)} \tilde{\mu}_i \nabla g_i(x^*, y^*)^\top d + \sum_{i\in\beta(x^*,y^*)} \tilde{\nu}_i \nabla \tau_i(x^*, y^*)^\top d = 0.$$

Since $\nabla g_i(x^*, y^*)^\top d > 0$ and $\nabla \tau_i(x^*, y^*)^\top d > 0$ for all $i \in \beta(x^*, y^*)$, the above equality holds if and only if $\tilde{\mu}_i = \tilde{\nu}_i = 0$ for all $i \in \beta(x^*, y^*)$. This leads to a contradiction. So $\{a_t\}_{t=1}^{\infty}$ must have a convergent subsequence.

Let $(\lambda^*, \hat{\mu}^*, \hat{\nu}^*)$ denote an accumulation point of $\{a_t\}_{t=1}^{\infty}$. As $t \to \infty$, (A.1) implies

$$\nabla F(x^*, y^*) - \nabla \phi(x^*, y^*)^\top \lambda^* - \nabla g(x^*, y^*)^\top \hat{\mu}^* - \nabla \tau(x^*, y^*)^\top \hat{\nu}^* = 0.$$

For every index $i \in \alpha(x^*, y^*)$, we have $g_i(x^*, y^*) = 0$ by definition and $\nu_i^{(t)} = 0$ for all t by complementarity conditions. Then $\hat{\nu}_i^{(t)} \to 0 = \hat{\nu}_i^*$ as $t \to 0$ by the continuity of τ_i. Similarly, it holds that

$$\hat{\mu}_j^* = 0, \ \forall j \in \gamma(x^*, y^*); \quad \hat{\mu}_k^*, \hat{\nu}_k^* \geq 0, \ \forall k \in \beta(x^*, y^*).$$

Therefore, (x^*, y^*) is an S-stationary point of (2.4).

(ii). Denote $h_t := g(x^{(t)}, y^{(t)})^\top \tau(x^{(t)}, y^{(t)})$. We first show that the sequence $\{h_t\}_{t=1}^{\infty}$ is non-increasing. For every $t' > t$, since $(x^{(t)}, y^{(t)})$ is a global solution of (3.2) with $\theta = \theta_t$, we have

$$\begin{aligned} F(x^{(t')}, y^{(t')}) + \theta_t h_{t'} &\geq F(x^{(t)}, y^{(t)}) + \theta_t h_t, \\ F(x^{(t)}, y^{(t)}) + \theta_{t'} h_t &\geq F(x^{(t')}, y^{(t')}) + \theta_{t'} h_{t'}. \end{aligned}$$

Adding the two inequalities, we get $(\theta_{t'} - \theta_t)(h_t - h_{t'}) \geq 0$. Since $\theta_{t'} - \theta_t \geq 0$ by construction, $h_t \geq h_{t'}$ and thus $\{h_t\}_{t=1}^{\infty}$ is a non-increasing sequence.

Next, we prove that $\lim_{t\to\infty} h_t = 0$. Suppose by contradiction that there exists an $\epsilon > 0$ and $T \in \mathbb{N}$ such that $h_t > \epsilon$ for all $t > T$. Let $(\bar{x}, \bar{y})$ be an arbitrary feasible point of (3.2) and $(\tilde{x}, \tilde{y})$ the global optimizer of $F(x, y)$ without constraints. For all t large enough, we have $h_t > \epsilon$ and

$$\theta_t \geq \frac{1}{\epsilon}|F(\bar{x}, \bar{y}) - F(\tilde{x}, \tilde{y})| + 2.$$

The above inequality implies that

$$F(\tilde{x}, \tilde{y}) + \theta_t \epsilon \geq F(\tilde{x}, \tilde{y}) + |F(\bar{x}, \bar{y}) - F(\tilde{x}, \tilde{y})| + 2\epsilon > F(\bar{x}, \bar{y}).$$

Since $h_t > \epsilon$, we further have

$$F(\bar{x}, \bar{y}) \leq F(\tilde{x}, \tilde{y}) + \theta_t h_t \leq F(x^{(t)}, y^{(t)}) + \theta_t h_t.$$

Since the feasible set of (3.2) is contained within that of (3.5) and their objectives are equivalent on this shared set, the optimal value of (3.5) provides a lower bound for (3.2). In other words,

$$F(\bar{x}, \bar{y}) \leq F(x^{(t)}, y^{(t)}) + \theta_t h_t \leq \inf\{F(x, y) : \phi(x, y) = 0, \, 0 \leq g(x, y) \perp \tau(x, y) \geq 0\},$$

which contradicts to the fact that $(\bar{x}, \bar{y})$ is a feasible point of (3.2). Therefore, we have $\lim_{t\to\infty} h_t = 0$. Since $g(x, y)$ and $\tau(x, y)$ are continuous functions, we have $g(x^*, y^*)^\top \tau(x^*, y^*) = 0$, indicating the result.