Second-Order Sensitivity Analysis for Bilevel Optimization

Robert Dyro
Stanford University
rdyro@stanford.edu

Edward Schmerling
Stanford University
schmrlng@stanford.edu

Nikos Aréchiga
Toyota Research Institute
nikos.arechiga@tri.global

Marco Pavone
Stanford University
pavone@stanford.edu

Abstract

In this work we derive a second-order approach to bilevel optimization, a type of mathematical programming in which the solution to a parameterized optimization problem (the "lower" problem) is itself to be optimized (in the "upper" problem) as a function of the parameters. Many existing approaches to bilevel optimization employ first-order sensitivity analysis, based on the implicit function theorem (IFT), for the lower problem to derive a gradient of the lower problem solution with respect to its parameters; this IFT gradient is then used in a first-order optimization method for the upper problem. This paper extends this sensitivity analysis to provide second-order derivative information of the lower problem (which we call the IFT Hessian), enabling the usage of faster-converging second-order optimization methods at the upper level. Our analysis shows that (i) much of the computation already used to produce the IFT gradient can be reused for the IFT Hessian, (ii) errors bounds derived for the IFT gradient readily apply to the IFT Hessian, (iii) computing IFT Hessians can significantly reduce overall computation by extracting more information from each lower level solve. We corroborate our findings and demonstrate the broad range of applications of our method by applying it to problem instances of least squares hyperparameter auto-tuning, multi-class SVM auto-tuning, and inverse optimal control.

1 INTRODUCTION

Optimization is the foundation of modern learning and decision making systems, therefore a natural problem of interest is how to improve, learn, or optimize the optimization itself. Many practitioners of autonomous driving, robotics, and machine learning employ optimization on an everyday basis. Understanding how best to adjust this tool to more accurately suit their application needs is key to improving performance, trust, and reliability of these systems.

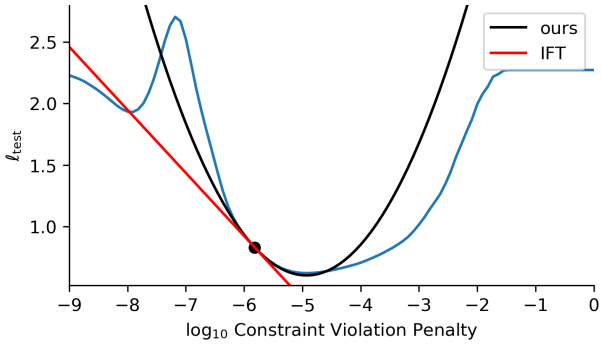


Figure 1: The single (hyper-)parameter test loss landscape of a multi-class SVM on Fashion-MNIST. Evaluating a point on this curve takes ∼100 seconds. We obtain a local quadratic approximation which leads to a much faster (hyper-)parameter optimization.

A natural way to approach improving optimization comes through formulating a bilevel program—users solve an optimization problem constructed with given data and parameters, and rely on a secondary metric quantifying the quality of the optimization result (it is of course optimal with respect to its own objective) to inform system design. Examples include (i) tuning regularization of a regression model to give good results on the test set, while training it to optimality on the train set, (ii) designing an autonomous car that drives in a human-like fashion, where it optimizes a finite horizon trajectory planning problem at every time step, (iii) setting parallel auction prices in such a way that rational bidding (an optimization in itself) leads to highest auction holder revenue. Any time optimization or decision making is applied, the question of selecting the right (hyper-)parameters arises in order to obtain, for example, (i) statistical models which generalize better, (ii) autonomous agents that behave more like an expert, (iii) auction systems that cannot be exploited. This notion is formalized as a bilevel

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

program in which the *optimization-to-be-improved* represents the lower level:

$$\begin{aligned} \underset{p}{\text{minimize}} \quad & f_U(z^\star, p) \\ \text{subject to} \quad & z^\star = \underset{z \in \mathbb{Z}}{\operatorname{argmin}}\, f_L(z, p). \end{aligned} \tag{1}$$

This bilevel program formulation is general and subsumes the problems of test set model generalization, Stackelberg competition (Von Stackelberg, 2010), meta-learning (Finn et al., 2017) and few-shot learning (Lee et al., 2019). The quality of the optimization result, $z^\star$, of the objective f_L is quantified via the upper objective, f_U. In the example of a statistical model, f_L represents the loss on the train dataset and f_U the loss on the test dataset. Solving the bilevel program requires selecting parameters p which produce such $z^\star$ that together lead to the minimal upper level loss f_U.[1]

This solution is often approximated by selecting parameters p by hand or via grid search. However, these approaches suffer from (a) being limited to cases where the dimension of p is low (usually below 4), (b) requiring parallel computing resources to keep re-evaluating $z^\star(p)$ and most critically, (c) the search is often done manually and wastes the expert's or practitioner's time.

A more principled way of solving Problem (1) is to use derivative information and make use of general-purpose solvers developed for optimization problems. This, however, requires the derivative of $z^\star$ with respect to p—quantifying how small changes in p affect the upper level objective f_U not just directly, but also by influencing $z^\star$. Although a closed-form expression of $z^\star$ with respect to p rarely exists, because $z^\star$ is the result of optimization, the dependence is *implicitly* defined via the necessary conditions for optimality of the lower optimization. For smooth-in-parameter problems, the implicit function theorem (IFT) may be employed to compute this derivative information relevant to solving the upper problem.

The availability of gradient expressions in bilevel programming obtained via sensitivity analysis enables the use of existing powerful optimizer to tackle these problems when they arise in real-world applications. However, even though many optimization algorithms use second-order information to converge fast in cases where the forward function evaluation is the bottleneck, not much attention has been paid to extending sensitivity analysis to second-order information. Doing so would enable another class of faster optimization algorithms to be applied to bilevel programming.

[1] When $z^\star$ itself has an interpretation as "parameters", e.g., in model learning, p may be referred to as "hyperparameters"; we will refer to p as parameters throughout the remainder of this domain-agnostic work.

The theoretical application of sensitivity analysis to bilevel programming relies on exact solutions to the lower level problems, but in reality, the numerical limitations rarely allow for that. Existing literature on first-order methods thus focuses on showing that the error in the derivative can be bounded and goes to zero as the approximation approaches the solution to the lower level problem.

1.1 Contributions

In this work we extend the application of the implicit function theorem, where *gradients* of inner optimization result with respect to parameters are found as in many existing works, (Gould et al., 2016; Agrawal et al., 2019b; Barratt, 2018), and derive second-order derivatives, i.e., the IFT Hessian. We leverage this result for three main contributions: (i) We show that the computational complexity of obtaining second-order derivatives is, in many cases, still dominated by the same matrix inversion bottleneck required for the IFT gradient and so our method can be implemented equally efficiently. (ii) We analyze our IFT Hessian expression to derive computational complexity and error bound expressions. We derive a new form of the regularized error bound under diagonal regularization of the matrix inverse operation in the application of IFT. (iii) We use our second-order derivative expression to apply second-order optimization methods to two machine learning datasets and show that these methods lead to faster bilevel optimization, requiring fewer lower level problem evaluations. We then further discuss the practical limitations and advantages of second-order optimization for bilevel optimization.

We open-source our implementation in two popular machine learning/scientific computing/automatic differentiation frameworks, PyTorch[2] and JAX[3], in a user-friendly format at https://github.com/StanfordASL/sensitivity_torch and https://github.com/StanfordASL/sensitivity_jax.

1.2 Related Work

Practical Deployment Optimization improvement or optimization tuning has a long history in practical applications. For systems for which gradient derivation is non-trivial or more generally for systems where local gradient information is not informative of the global scope of the problem, gradient-free proxy models may be employed as in Golovin et al. (2017). Like hand-tuning or grid search, this approach constrains the number of parameters that can practically be tuned to single or low double digits.

[2] pytorch.org

[3] github.com/google/jax

Formal Literature More formally the Bilevel Programming Problem (BLPP) formulation has a long history in literature (Bard, 1984; Von Stackelberg, 2010; Henrion and Surowiec, 2011; Liu et al., 2001; Bard, 2013). Sinha et al. (2017) contains an extensive review of approaches to solving BLPPs. Many of these works focus on theoretical analysis/characterization of BLPP approaches; the focus of this paper is more on specific concrete applications.

Applications-oriented Renewed interest in applications-oriented gradient-based solutions to BLPPs (Bengio, 2000), led to several works establishing the techniques for obtaining lower level solution gradients with respect to the parameters (Gould et al., 2016) and doing so efficiently for convex problems (Barratt, 2018; Agrawal et al., 2019b,a). Several computationally optimized, program-form specific approaches have been shown (Amos and Kolter, 2017; Amos et al., 2018). Most recently Lorraine et al. (2020); Blondel et al. (2021) apply the techniques to large-scale programs. Most of these applications-oriented works focus on deriving gradient expressions—to be used with a gradient-only BLPP optimizer. These works do not consider the loss landscape or the local curvature of the BLPP; in contrast, in this work, we attempt to quantify that.

Machine Learning BLPPs also found applications in meta-learning literature (Andrychowicz et al., 2016; Finn et al., 2017; Harrison et al., 2018; Bertinetto et al., 2018) with several works making explicit use of the implicit function theorem (Lee et al., 2019; Rajeswaran et al., 2019). While meta-learning literature poses an important application for BLPP, so far little attention has been given to improving the specifics of the solution methods employed in this body of work.

Higher-Order Derivatives Works most closely related to ours, with a focus on finding higher derivative information and analyzing the curvature of the BLPP are Wachsmuth (2014); Mehlitz and Zemkoho (2021). These works do not demonstrate how to efficiently compute second order derivatives or demonstrate their usage in Newton's-Method-like optimization, two key focuses of our work.

1.3 Notation

For an argument $x \in \mathbb{R}^d$ we denote the dimension of x by $\dim(x) = d$. For a scalar function $f : \mathbb{R}^{\dim(x)} \to \mathbb{R}$ we denote its gradient and Hessian as $\nabla_x f(x)$ and $\nabla_x^2 f(x)$. For a vector function of $g : \mathbb{R}^{\dim(x)} \to \mathbb{R}^{\dim(g)}$ we denote its Jacobian matrix as $D_x g(x) \in \mathbb{R}^{\dim(g)\times\dim(x)}$. Where a function takes two arguments, we use the normal and mono-space font respectively to denote whether the differentiation operator is partial or total (i.e. does the derivative capture dependence between variables), e.g. for a function $g : \mathbb{R}^{\dim(x)} \times \mathbb{R}^{\dim(y)} \to \mathbb{R}^{\dim(g)}$, $D_x g(x, y)$ denotes the partial Jacobian of g w.r.t. x and $\mathtt{D}_x g(x)$ the total Jacobian of g w.r.t. x accounting for possible dependence of x on y. In this work, we aim to describe higher order derivatives of vector functions, which leads us to define, for $g : \mathbb{R}^{\dim(x)} \times \mathbb{R}^{\dim(y)} \to \mathbb{R}^{\dim(g)}$, $D_{xy} g \in \mathbb{R}^{\dim(g)\dim(x)\times\dim(y)}$, represented as a two dimensional matrix s.t.

$$D_{xy} g(x, y) = \begin{bmatrix} D_y\big(\nabla_x \left(g(x, y)_1\right)\big) \\ \vdots \\ D_y\big(\nabla_x \left(g(x, y)_{\dim(g)}\right)\big) \end{bmatrix} \tag{2}$$

where $g(x, y)_i$ denotes the i-th scalar output of the vector function g. We define the operator $H_x \equiv D_{xx}$ (and the total version $\mathtt{H}_x \equiv \mathtt{D}_{xx}$). We use $\otimes$ to denote the Kronecker product.

2 PROBLEM STATEMENT

Recall the formulation of bilevel programming:

$$\begin{aligned} \underset{p}{\text{minimize}} \quad & f_U(z^\star, p) \\ \text{subject to} \quad & z^\star = \underset{z\in\mathbb{Z}}{\operatorname{argmin}}\, f_L(z, p). \end{aligned} \tag{1}$$

The problem considered in this work is how to obtain an explicit expression for the total second-order derivative, the Hessian matrix, of f_U with respect to p, i.e., $\mathtt{H}_p f_U(z^\star, p)$. Access to this Hessian enables the application of second-order optimization methods for solving the upper problem, with the aim of reducing the total number of lower problem optimizations (and correspondingly, total overall computation).

3 METHOD

3.1 Preliminaries

Necessary Derivatives As overviewed in Section 1.2, there are several approaches to solving the bilevel problem (1). Here, we focus on derivation of the first and second derivatives of f_U w.r.t. p: $\mathtt{D}_p f_U$ and $\mathtt{H}_p f_U$. Because $z^\star$ depends on p, the total derivatives of f_U can be written as

$$\mathtt{D}_p f_U(z^\star, p) = D_p f_U + \big(D_{z^\star} f_U\big)\big(D_p z^\star\big) \tag{3}$$

$$\begin{aligned} \mathtt{H}_p f_U(z^\star, p) = {} & H_p f_U + \big(D_p z^\star\big)^T \big(H_{z^\star} f_U\big)\big(D_p z^\star\big) \\ & + \Big(\big(D_{z^\star} f_U\big) \otimes I\Big) H_p z^\star \end{aligned} \tag{4}$$

Importantly, Equations (3) and (4) depend on (i) terms *directly obtainable* from the upper objective

function via analytical or automatic differentiation: $D_p f_U$, $H_p f_U$, $D_{z^\star} f_U$, $H_{z^\star} f_U$ and (ii) terms quantifying the sensitivity of lower level optimization, i.e., the result $z^\star$ w.r.t. p: $D_p z^\star$, $H_p z^\star$.

The latter two terms are nominally obtainable through automatic differentiation by unrolling the entire f_L optimization process—in many cases this is computationally undesirable or simply too memory intensive to be feasible. An alternative way of obtaining optimization sensitivity terms follows from the IFT.

Implicit Function The main insight which allows differentiating $z^\star = \operatorname{argmin}_z f_L(z, p)$ is the implicit condition imposed on $z^\star$ as a result of $z^\star$ being an optimal solution. An optimal solution must satisfy First Order Optimality Conditions (FOOC)

$$k(z^\star, p) = 0 \tag{5}$$

which define an implicit equation for $z^\star$.[4] For unconstrained optimization, the FOOC we consider are explicitly:

$$k(z^\star, p) = D_{z^\star} f_L(z^\star, p) = 0.$$

The following first-order sensitivity analysis is standard in the literature; we reproduce it here to motivate and enable our second-order analysis.

3.2 Implicit Function Theorem (IFT)

Theorem 1 (Implicit Function Theorem (IFT))**.** *Let $k : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ be a continuously differentiable multivariate function of two variables $z^\star \in \mathbb{R}^m$ and $p \in \mathbb{R}^n$ such that k defines a fixed point for $z^\star$, i.e., $k(z^\star, p) = 0$ for all values of $p \in \mathbb{R}^n$. Then, the derivative of (implicitly defined) $z^\star$ w.r.t. p is given by*

$$D_p z^\star = -\big(D_{z^\star} k(z^\star, p)\big)^{-1} D_p k(z^\star, p).$$

Proof. Take $k(z^\star, p) = 0$ and apply the chain rule to differentiate w.r.t. p, then, if $D_{z^\star} k(z^\star, p)$ is invertible

$$\begin{aligned} k(z^\star, p) &= 0 \\ D_p k(z^\star, p) + \big(D_{z^\star} k(z^\star, p)\big)\big(D_p z^\star\big) &= 0 \\ -\big(D_{z^\star} k(z^\star, p)\big)^{-1}\big(D_p k(z^\star, p)\big) &= D_p z^\star. \end{aligned} \tag{6}$$

□

Understanding the proof of the equation above is vital to obtaining higher derivatives via the IFT, since under suitable smoothness assumptions, Equation (6) can be differentiated again and the resulting expression solved for $H_p z^\star$.

We now present the rarely derived IFT Hessian, which allows us to obtain the second derivative through an optimization. This result follows from a repeated application of the IFT to the same implicit function to obtain higher derivatives of $z^\star$ w.r.t. p.

3.3 Second-Order IFT

Theorem 2 (Second-Order IFT)**.** *Let $k : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ be a twice continuously differentiable multivariate function of two variables $z^\star \in \mathbb{R}^m$ and $p \in \mathbb{R}^n$ such that k defines a fixed point for x, i.e. $k(z^\star, p) = 0$ for all values of $p \in \mathbb{R}^n$. Then, the Hessian of (implicitly defined) $z^\star$ w.r.t. p is given by*

$$\begin{aligned} H_p z^\star = -\Big[\big(D_{z^\star} k\big)^{-1} \otimes I\Big]\Big[H_p k + \big(D_{pz^\star} k\big)\big(D_p z^\star\big) \\ + \big(I \otimes \big(D_p z^\star\big)^T\big)\big(D_{z^\star p} k\big) \\ + \big(I \otimes \big(D_p z^\star\big)^T\big)\big(H_{z^\star} k\big)\big(D_p z^\star\big)\Big]. \end{aligned} \tag{7}$$

The proof of Theorem 2 is provided in the Appendix.

4 ANALYSIS

4.1 Computational Complexity

The expression for the Second-Order IFT features Kronecker product terms (denoted by $\otimes$) which serve to broadcast over dimensions of either the embedding $\dim(z)$ or the parameter $\dim(p)$. Efficiently broadcasting over a particular dimension does not require constructing full dense matrices.

The two broadcasting Kronecker product-based forms that appear in Equation (4) are of the form $A \otimes I$ and $I \otimes B$ which both correspond to a broadcasted version of matrix multiplication. We refer the interested reader to the Appendix for the discussion of how computation with these forms can be accomplished efficiently.

Since our analysis here focuses on sensitivity analysis for optimization, we devote most of our attention to analyzing the computational complexity of computing the expression in Equation (4), which we recall here in full

$$\begin{aligned} \mathtt{H}_p f_U(z^\star, p) = H_p f_U + \big(D_p z^\star\big)^T \big(H_{z^\star} f_U\big)\big(D_p z^\star\big) \\ + \big(\big(D_{z^\star} f_U\big) \otimes I\big) H_p z^\star \end{aligned}$$

where $H_p f_U$, $H_{z^\star} f_U$ and $D_{z^\star} f_U$ have explicit expressions and the term $D_p z^\star$ is either already available

[4] An *implicit function* is simply defined as a function $k : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ that equals 0 if its first input x satisfies the implicit definition: $x \in \{x \mid k(x, y) = 0,\ \exists y \in \mathbb{R}^n\}$. This is in contrast to an *explicit function* of the form $f : \mathbb{R}^n \to \mathbb{R}^m$: $x \in \{x \mid f(y) = x\}$.

to us from first-order analysis or can be computed cheaply since the matrix $D_{z^\star}k$ had to be factorized for first-order analysis. Thus, the only term requiring significant computation, substituting Equation (7), is

$$\big(\big(D_{z^\star}f_U\big)\otimes I\big)\underbrace{\Big[\big(D_{z^\star}k\big)^{-1}\otimes I\Big]\Big[\dots\Big]}_{\text{Equation (7)}} = \Big[\Big(D_{z^\star}f_U\big(D_{z^\star}k\big)^{-1}\Big)\otimes I\Big]\Big[\dots\Big]$$

Since the upper level objective is a scalar by definition, the product

$$D_{z^\star}f_U\big(D_{z^\star}k\big)^{-1} = v^T \in \mathbb{R}^{1\times m} \tag{7}$$

can be computed at the cost of a single matrix solve using an already necessarily factorized matrix from first-order analysis, *which can be done computationally cheaply.* Thus, evaluating the term $\big(\big(D_{z^\star}f_U\big)\otimes I\big)H_p z^\star$ reduces to (a) caching a term from first-order analysis and (b) broadcasted vector-matrix product (a weighted summation of matrices). Alternatively, if an automatic differentiation system is used to compute the right bracket in Equation (7), then v can be used as a sensitivity vector in vector-Jacobian or Jacobian-vector products, significantly reducing the number of calls to the automatic differentiation (autodiff) engine.

4.1.1 Big-$\mathcal{O}$ Notation

Operation	Computational Complexity
1st IFT	$\mathcal{O}\big(m^3 + mn\big)$
1st IFT w/ sens.	$\mathcal{O}\big(m^3 + n\big)$
2nd IFT	$\mathcal{O}\big(m^3 + m^2n^2\big)$
2nd IFT w/ sens.	$\mathcal{O}\big(m^3 + mn^2\big)$

Table 1: Computational complexity of applying the first- and second-order implicit function theorem, omitting computation of partial explicit derivatives. In this table we define $m = \dim(z)$, $n = \dim(p)$. "w/ sens." denotes that a sensitivity vector is available in which case the matrix inverse can be computed for a single left hand side as in Equation (7).

Following Section 4.1 we show computational complexity of applying the first- (1st) and second-order (2nd) implicit function theorem in Table 1 and observe that the second-order expression with a sensitivity left-hand side—the term $D_{z^\star}f_U$ in Equations (3) (4)—has a computational complexity that differs from first-order expression with a sensitivity left hand side only by the additional quadratic term in $n = \dim(p)$, which is expected as the resulting matrix is in $\mathbb{R}^{n\times n}$. The increase in computational complexity is thus minor. Here we use the computational complexity of a matrix factorization operation to be $\mathcal{O}\big(m^3\big)$.

We do not include the computational complexity of obtaining the partial derivatives of f_U and k, which we denote as *partial explicit derivatives*, primarily because: they are heavily problem dependent; they vanish; can be precomputed; efficient analytical expressions exist; or they can be obtained by efficiently making use of Jacobian-vector and vector-Jacobian products in an automatic differentiation engine.

4.2 Error Analysis

Error analysis is vital for sensitivity analysis because numerical limitations often result in lower level solutions that are not quite optimal, but are instead at a small distance from the optimum. Developing confidence in the sensitivity methods described here requires the ability to bound the error caused by applying sensitivity analysis developed for optimal points to suboptimal lower level solutions.

4.2.1 First-Order Error Bound

Assuming an inexact local solution, we state the following error bound on the Jacobian. Theorem 3 is heavily inspired by Blondel et al. (2021), Theorem 1, which is in turn inspired by Higham (2002), Theorem 7.2.

Theorem 3 (First-Order Error Bound)**.** *Given the result of IFT applied to an exact solution $D_{z^\star}p = g = -A^{-1}B$, where $A(z^\star,p) = D_{z^\star}k(z^\star,p)$, $B(z^\star,p) = D_pk(z^\star,p)$ and an inexact solution $\widetilde{A}(z,p) = D_zk(z,p)$ (where $k(z,p)\neq 0$) and $\widetilde{B}(z,p) = D_pk(z,p)$. Assume $||z-z^\star|| \leq \delta$, $\Big|\Big|\widetilde{A}-A\Big|\Big|_{op} \leq \gamma\delta$, $\Big|\Big|\widetilde{B}-B\Big|\Big|_F \leq \beta\delta$, $||B||_F \leq R$, $\Big|\Big|\widetilde{A}v\Big|\Big| \geq \alpha_1||v||$, $||Av|| \geq \alpha_2||v||$, then*

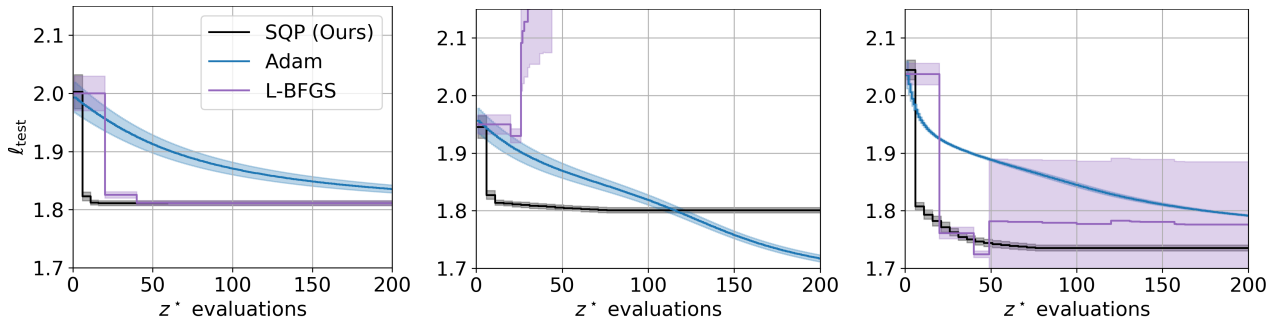
$$\Big|\Big|\widetilde{J}-J\Big|\Big|_F \leq \frac{\beta}{\alpha_1}\delta + \frac{\gamma R}{\alpha_1\alpha_2}\delta \tag{8}$$

We point the reader to the Appendix for the proof.

4.2.2 Second-Order Error Bound

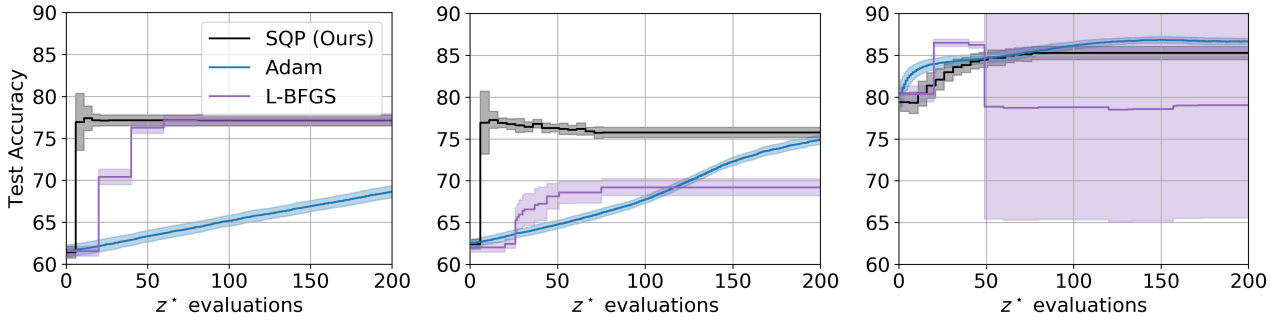
Assuming an inexact local solution, we state the following error bound on the Jacobian.

Theorem 4 (Second-Order Error Bound)**.** *Given the result of IFT applied to an exact solution, for $z^\star \in \mathbb{R}^m$, $H_{z^\star}p = H = -A^{-1}B$, where $A(z^\star,p) = D_{z^\star}k(z^\star,p)\otimes I$, $B(z^\star,p) = H_pk + \big(D_{pz^\star}k\big)\big(D_pz^\star\big) + \big(I\otimes\big(D_pz^\star\big)^T\big)\big(D_{z^\star p}k\big) + \big(I\otimes\big(D_pz^\star\big)^T\big)\big(H_{z^\star}k\big)\big(D_pz^\star\big)$ and an inexact solution*



(a) `RR` (b) `diag` (c) `conv`

Figure 2: Hyperparameter optimization of least-squares models with two gradient-only algorithms and one gradient & Hessian—enabled by this work.



(a) `RR` (b) `diag` (c) `conv`

Figure 3: Hyperparameter optimization of least-squares models with two gradient-only algorithms and one gradient & Hessian—enabled by this work.

$\widetilde{A}(z,p) \neq 0$ *and* $\widetilde{B}(z,p)$. *Assume* $||z - z^\star|| \leq \delta$, $\left|\left|\widetilde{A} - A\right|\right|_{op} \leq \gamma\delta$, $||H_p k(z,p) - H_p k(z^\star,p)||_F \leq \zeta\delta$, $||D_{zp}k(z,p) - D_{z^\star p}k(z^\star,p)||_F \leq \eta\delta$, $||H_z k(z,p) - H_{z^\star}k(z^\star,p)||_F \leq \nu\delta$, $||D_z p - D_{z^\star}p||_F \leq \kappa_J\delta$ *defined in Theorem 3,* $\left|\left|\widetilde{A}v\right|\right| \geq \alpha_1||v||$, $||Av|| \geq \alpha_2||v||$, *then*

$$\left|\left|\widetilde{H} - H\right|\right|_F \leq \frac{\zeta + 2\eta\kappa_g + \nu\kappa_g^2}{\alpha_1}\delta + \frac{\gamma R_H}{\alpha_1\alpha_2}\delta. \quad (9)$$

We point the reader to the Appendix for the proof.

4.2.3 Error Bound Optimization

Now, assuming an inexact solution we analyze the diagonal regularization of the inverse in the application of IFT. We let $\widehat{A} = \widetilde{A} + \epsilon I$.

Theorem 5 (Regularized First-Order Error Bound)**.** *Given the result of regularized IFT applied to an exact solution* $D_{z^\star}p = g = -A^{-1}B$, *where* $A(z^\star,p) = D_{z^\star}k(z^\star,p)$, $B(z^\star,p) = D_p k(z^\star,p)$ *and an inexact solution* $\widetilde{A}(z,p) = D_z k(z,p) + \epsilon I$, $\widetilde{B}(z,p) = D_p k(z,p)$ *(where* $k(z,p) \neq 0$*) . Assume* $||z - z^\star|| \leq \delta$, $\left|\left|\widetilde{A} - A\right|\right|_{op} \leq \gamma\delta$, $\left|\left|\widetilde{B} - B\right|\right|_F \leq \beta\delta$, $||B||_F \leq R$, $\left|\left|\widetilde{A}v\right|\right| \geq \alpha_1||v||$ *and* $v^T\widetilde{A}v \geq 0$, *so* $\left|\left|\widehat{A}v\right|\right| \geq (\alpha_1 + \epsilon)||v||$, $||Av|| \geq \alpha_2||v||$ *then*

$$\left|\left|\widehat{J} - J\right|\right|_F \leq \frac{\beta\delta}{\alpha_1 + \epsilon} + \frac{R\big(\gamma\delta + \epsilon\big)}{\big(\alpha_1 + \epsilon\big)\alpha_2} \quad (10)$$

Selecting a *post hoc* arbitrary regularization allows to tighten the bound. We refer the reader to the Appendix for the proof.

5 EXPERIMENTS

5.1 Regression Model Auto-Tuning

We compare the performance of three commonly used nonlinear optimization algorithms on the problem of linear model improvement via smooth hyperparameter tuning. Using the above analysis we are able to apply a second-order optimization method, which offers to dramatically reduce the number of lower function evaluations in BLPP and significantly speed up optimization problems where the lower level constraint evaluation dominates. To compare, we optimize the hyperparameters of 3 linear models on MNIST (LeCun,

1998) using three commonly used optimization algorithms, two gradient-only: (i) Adam (Kingma and Ba, 2014), (ii) L-BFGS (Liu and Nocedal, 1989) and one using second-order information (enabled by this work) (iii) SQP (Nocedal and Wright, 2006). The three linear models we choose all employ a least-squares lower level loss where the target vector is the one-hot encoding of the ten MNIST digits. We train on 1000 randomly selected MNIST examples in the train set and evaluate the upper level (test) loss on all examples in the test set. The upper level loss is the cross-entropy classification loss, $f_U(z^\star, p) = -\sum_j^{N_{\text{test}}} \sum_{i=1}^{10} t_{i,j} \log\left(q_{i,j}\right)$ where $t_{i,j} = \delta_{i,y_j}$ and $q_{i,j} = e^{x_j^T z^\star{}_i} / \left(\sum_{i=1}^{10} e^{x_j^T z^\star{}_i}\right)$.[5] For all models $z \in \mathbb{R}^{nd}$ where $n = 10$ is the number of MNIST classes and d is the number of features in the data vector.

Model 1 (`RR`) A single hyperparameter least-squares model with Tikhonov regularization (Tikhonov, 1943), also known as ridge regression (Gruber, 2017). The lower level loss takes the form $f_L(z, p) = ||Xz - Y||_{\text{F}}^2 + 10^p ||z||_{\text{F}}^2$. The features are raw image pixels and a bias term.

Model 2 (`diag`) A least-squares model with Tikhonov regularization where each weight is penalized with a separate weight. The lower level loss takes the form $f_L(z, p) = ||Xz - Y||_{\text{F}}^2 + \sum_i^{\dim(z)} 10^{p_i} z_i^2$. The features are the same as in Model 1.

Model 3 (`conv`) A least-squares model where the images are first passed through a parametric 2D convolution filter with a 3×3 kernel, a stride of 2, a bias, 1 input channel and 2 output channels. The convolution output is passed through the tanh activation function, before adding bias and applying the Tikhonov regularized least-squares model. The reduced images have a dimension of 338. The convolution weight, bias and the scalar Tikhonov regularization weight form the vector p.

We show the optimization results in Figures 2 & 3. The use of second-order derivative information significantly reduces the number of least-squares evaluations.

In `RR` our method, SQP, outperforms other optimizers both in terms of test accuracy and test loss. In `diag` L-BFGS goes unstable and the Adam optimizer tends to outperform SQP in terms of the test loss, but it takes a 100 evaluations and the SQP converges much quicker to a high test accuracy. Finally, the L-BFGS also exhibits poor performance on the `conv` model, SQP converges to a lower classification loss much faster. Adam reaches a higher test accuracy, but also a higher classification loss.

[5] $t_{i,j} = \delta_{i,y_j}$ indicates that $t_{i,j}$ is equal to 1 if example j is of the class i and 0 otherwise.

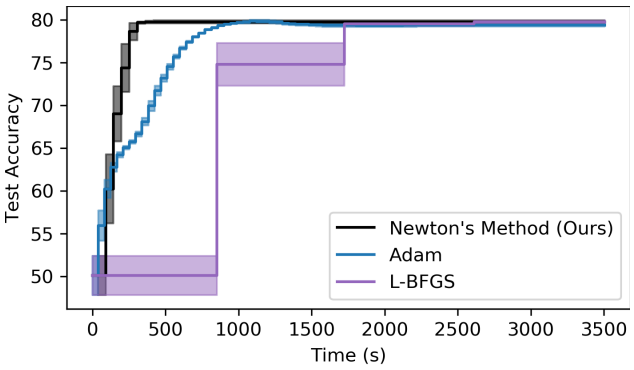


Figure 4: Optimization of multi-class SVM on Fashion-MNIST. Newton's method using second-order derivatives derived in this paper leads to much faster convergence.

5.2 Hyperparameter Optimization

We further verify the usefulness of computing second-order derivatives in practical problem instances by applying the Newton's Method to tuning a multi-class SVM model (Crammer and Singer, 2001) on the Fashion-MNIST (Xiao et al., 2017) dataset for best performance on the test set. We reformulate the constraints as log-barrier penalty[6] to ensure smoothness, with the refinement value of $\alpha = 10^2$. We solve the resulting problem with the Mosek optimizer.[7]

We scale the images down to 14×14 to aid with the memory requirements and fit a multi-class SVM model to 2,000 randomly selected samples in the train set. We define upper level loss as the cross-entropy classification loss between the predictions and the labels. We select 5 random seeds to verify algorithmic performance.

We show a test accuracy plot in Figure 4. We note that the optimization time, i.e., training the lower level multi-class SVM, dominates both the gradient and Hessian computation. The application of Newton's Method (enabled by second-order derivatives shown in this work) significantly reduces the number of necessary lower level optimizations and reaches maximum test accuracy much quicker in terms of wall-clock time.

[6] The log-barrier is defined for a constraint $x \leq 0$ as $-\log\left(-\alpha x\right)/\alpha$ for a tunable refinement constant α.

[7] www.mosek.com

5.3 Parameter Loss Landscape in Inverse Optimal Control

We investigate the Inverse Optimal Control (IOC) problem as a case study in loss landscape or curvature analysis. IOC has a natural formulation as a BLPP.

IOC has a large body of literature of its own, e.g. Keshavarz et al. (2011); Johnson et al. (2013); Terekhov and Zatsiorsky (2011), but we focus here on some simple examples that are illustrative to the general BLPP curvature analysis. We show that the unconstrained Optimal Control Problem (OCP) problem we formulate is globally convex in parameter, but that in the presence of constraints in the OCP, the resulting BLPP requires more care—linear inequalities, without reformulation, violate our smoothness assumption in Theorem 2.

Given known linear time-invariant discrete time dynamics $x^{(i+1)} = Ax^{(i)} + Bu^{(i)}$ and state and control cost matrices Q, R, and control limits $u^{(i)} \in \mathbb{U} = \{u \mid ||u||_\infty \leq u_{\text{lim}}\}$, we assume we observe some expert's trajectory $X_\text{e} = [x^{(0)}, \dots, x^{(N)}]$ and control history $U_\text{e} = [u^{(0)}, \dots, u^{(N)}]$. We seek to learn a reference trajectory we assume the expert is tracking, i.e., the expert behaves optimally under the cost $J(X, U)$:

$$X_\text{e}, U_\text{e} = \underset{x^{(i+1)}=Ax^{(i)}+Bu^{(i)}, U\in\mathbb{U}}{\text{argmin}} \; J(X, U, X_\text{e}, U_\text{e}) \quad (11)$$

where $J(X, U, X_\text{e}, U_\text{e}) = \sum_i (x^{(i)} - x^{(i)}_{\text{e,ref}})^T Q (x^{(i)} - x^{(i)}_{\text{e,ref}}) + (u^{(i)} - u^{(i)}_{\text{e,ref}})^T R (u^{(i)} - u^{(i)}_{\text{e,ref}})$. This leads to a BLPP

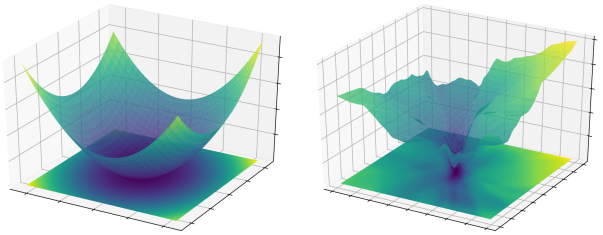
$$\begin{aligned} &\min_{X_{\text{ref}}, U_{\text{ref}}} \quad ||X_\text{e} - X^\star||_2^2 + ||U_\text{e} - U^\star|| \\ &\text{s.t.} \quad X^\star, U^\star = \underset{\substack{x^{(i+1)}=Ax^{(i)}+Bu^{(i)}, \\ U\in\mathbb{U}}}{\text{argmin}} \; J(X, U, X_{\text{ref}}, U_{\text{ref}}). \end{aligned} \quad (12)$$

In the typical notation of this paper, $z^\star = (X^\star, U^\star)$, $p = (X_{\text{ref}}, U_{\text{ref}})$. The problem corresponds to discovering the reference trajectory of an optimal agent, e.g., the centerline of a road using an observed trajectory of an autonomous car.

We show the comparison between the upper loss landscapes for IOC with unconstrained/constrained controls in Figure 5. We employ the Principle Component Analysis (PCA) dimension reduction technique on the optimization path to visualize a many dimensional upper loss in 2 dimensions, inspired by Li et al. (2017).

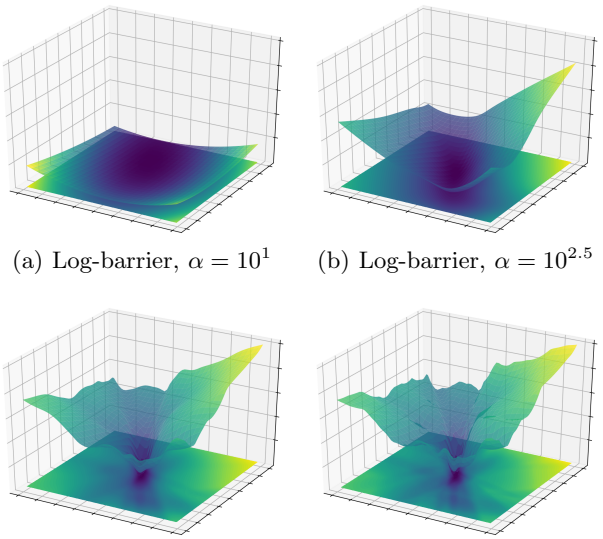
In the absence any constraints, the resulting problem as stated is convex which follows from the application of Equations (4) and (7).

Naive application of Equation (7) to a constrained IOC problem yields a globally positive definite Hessian, but violates the assumptions of Theorem 2, since FOOC with linear constraints are not twice continuously differentiable. In the presence of even the simple maximum control value constraints considered, the loss-in-parameter (the upper loss) is highly non-convex. We observe that any constraints can be smoothly approximated to any degree of precision using the log-barrier function which is infinitely differentiable. Figure 6 shows the 2D projected loss-in-parameter p landscape for a successive refinement of the log-barrier constraints; for high values of refinement $\alpha \to \infty$, the loss landscape closely approximates the exactly constrained version, yet remains differentiable, so Theorem 2 can be applied.



(a) Unconstrained (b) Control Constrained

Figure 5: Comparison of upper loss landscape in inverse optimal control without and with control constraints. Surface and the contour plots show the PCA 2D projections based on the optimization path.



(a) Log-barrier, $\alpha = 10^1$ (b) Log-barrier, $\alpha = 10^{2.5}$

(c) Log-barrier, $\alpha = 10^4$ (d) Exactly constrained

Figure 6: Comparison of loss landscape for inverse box-constrained-control MPC. The surface and contour plots show the PCA 2D projections based on the optimization path.

Code release The code for our experiments is contained at https://github.com/StanfordASL/Second-OrderSensitivityAnalysisForBilevelOptimization.

6 DISCUSSION

Limitations & Promises The method we propose here offers to make better use of every single lower level problem evaluation, but comes with limitations. Firstly, like any second-order optimization method, it might not be best suited for highly non-convex landscapes, which can be common in BLPPs as Figure 5 shows. Secondly, in computational complexity analysis we do not focus on obtaining partial explicit derivatives necessary to compute the second-order sensitivity expression. We do so because these can be problem specific and often be zero or have an analytic form, but if they are obtained via automatic differentiation (the method we employ), their computation time highly depends on chosen software package. In general, it might turn out that computing second-order sensitivity information might take longer than several evaluations of the lower level and the first-order sensitivity information evaluation—at which point gradient-only optimization methods will likely function better than our proposed approach. Nevertheless, our work expands the optimization toolbox where some examples of BLPPs can be optimized much quicker, as Figure 4 shows.

7 CONCLUSIONS

In this work we derive the second-order derivatives of the upper level objective in a general bilevel program via sensitivity analysis using the implicit function theorem. Second-order information enables second-order optimization to be applied to these problems, which we argue can drastically reduce the number of lower level function evaluations, and speed up optimization, in cases where the lower level evaluation dominates. We further show that the computational complexity of our proposed approach is comparable to first-order IFT and we adapt error bound analysis for first-order IFT derivatives to our second-order IFT derivatives.

Future Work Future work includes quantifying how well various approximations suggested for first-order IFT apply to second-order IFT and whether (and when), for optimization purposes, some terms in our second-order sensitivity expression can be omitted or approximated to make their computation quicker. Finally, we are interested in further investigating loss landscapes, and the associated difficulty of optimization, for constrained bilevel problems.

Acknowledgments

Toyota Research Institute and the NASA University Leadership Initiative (grant #80NSSC20M0163) provided funds to support this work; this article solely reflects the opinions and conclusions of its authors and not any Toyota or NASA entity.

References

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, Z. (2019a). Differentiable convex optimization layers. *arXiv preprint arXiv:1910.12430*.

Agrawal, A., Barratt, S., Boyd, S., Busseti, E., and Moursi, W. M. (2019b). Differentiating through a cone program. *arXiv preprint arXiv:1904.09043*.

Ailon, N. and Chazelle, B. (2006). Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 557–563.

Amos, B. and Kolter, J. Z. (2017). Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR.

Amos, B., Rodriguez, I. D. J., Sacks, J., Boots, B., and Kolter, J. Z. (2018). Differentiable mpc for end-to-end planning and control. *arXiv preprint arXiv:1810.13400*.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pages 3981–3989.

Bard, J. F. (1984). Optimality conditions for the bilevel programming problem. *Naval research logistics quarterly*, 31(1):13–26.

Bard, J. F. (2013). *Practical bilevel optimization: algorithms and applications*, volume 30. Springer Science & Business Media.

Barratt, S. (2018). On the differentiability of the solution to convex optimization problems. *arXiv preprint arXiv:1804.05098*.

Barratt, S. T. and Boyd, S. P. (2021). Least squares auto-tuning. *Engineering Optimization*, 53(5):789–810.

Bengio, Y. (2000). Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900.

Bertinetto, L., Henriques, J. F., Torr, P. H., and Vedaldi, A. (2018). Meta-learning with dif-

ferentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*.

Bertrand, Q., Klopfenstein, Q., Blondel, M., Vaiter, S., Gramfort, A., and Salmon, J. (2020). Implicit differentiation of lasso-type models for hyperparameter optimization. In *International Conference on Machine Learning*, pages 810–821. PMLR.

Blondel, M., Berthet, Q., Cuturi, M., Frostig, R., Hoyer, S., Llinares-López, F., Pedregosa, F., and Vert, J.-P. (2021). Efficient and modular implicit differentiation. *arXiv preprint arXiv:2105.15183*.

Crammer, K. and Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research*, 2(Dec):265–292.

Diamond, S. and Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5.

El Ghaoui, L. (2002). Inversion error, condition number, and approximate inverses of uncertain matrices. *Linear algebra and its applications*, 343:171–193.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR.

Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., and Sculley, D. (2017). Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1487–1495.

Golub, G. H. and Van Loan, C. F. (1996). Matrix computations. johns hopkins studies in the mathematical sciences.

Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R. S., and Guo, E. (2016). On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*.

Gruber, M. H. (2017). *Improving efficiency by shrinkage: the James-Stein and ridge regression estimators*. Routledge.

Harrison, J., Sharma, A., and Pavone, M. (2018). Meta-learning priors for efficient online bayesian regression. *arXiv preprint arXiv:1807.08912*.

Henrion, R. and Surowiec, T. (2011). On calmness conditions in convex bilevel programming. *Applicable Analysis*, 90(6):951–970.

Higham, N. J. (2002). *Accuracy and stability of numerical algorithms*. SIAM.

Johnson, M., Aghasadeghi, N., and Bretl, T. (2013). Inverse optimal control for deterministic continuous-time nonlinear systems. In *52nd IEEE Conference on Decision and Control*, pages 2906–2913. IEEE.

Keshavarz, A., Wang, Y., and Boyd, S. (2011). Imputing a convex objective function. In *2011 IEEE international symposium on intelligent control*, pages 613–619. IEEE.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

LeCun, Y. (1998). The mnist database of handwritten digits.

Lee, K., Maji, S., Ravichandran, A., and Soatto, S. (2019). Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10657–10665.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2017). Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*.

Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528.

Liu, G., Han, J., and Zhang, J. (2001). Exact penalty functions for convex bilevel programming problems. *Journal of Optimization Theory and Applications*, 110(3):621–643.

Lorraine, J., Vicol, P., and Duvenaud, D. (2020). Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1552. PMLR.

Magnus, J. R. and Neudecker, H. (2019). *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons.

Mehlitz, P. and Zemkoho, A. B. (2021). Sufficient optimality conditions in bilevel programming. *Mathematics of Operations Research*.

Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

Rajeswaran, A., Finn, C., Kakade, S., and Levine, S. (2019). Meta-learning with implicit gradients.

Sinha, A., Malo, P., and Deb, K. (2017). A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295.

Terekhov, A. V. and Zatsiorsky, V. M. (2011). Analytical and numerical analysis of inverse optimization problems: conditions of uniqueness and computational methods. *Biological cybernetics*, 104(1):75–93.

Tikhonov, A. N. (1943). On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pages 195–198.

Von Stackelberg, H. (2010). *Market structure and equilibrium*. Springer Science & Business Media.

Wachsmuth, G. (2014). Differentiability of implicit functions: Beyond the implicit function theorem. *Journal of Mathematical Analysis and Applications*, 414(1):259–272.

Wiesemann, W., Tsoukalas, A., Kleniati, P.-M., and Rustem, B. (2013). Pessimistic bilevel optimization. *SIAM Journal on Optimization*, 23(1):353–380.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Yao, Z., Gholami, A., Shen, S., Mustafa, M., Keutzer, K., and Mahoney, M. W. (2020). Adahessian: An adaptive second order optimizer for machine learning. *arXiv preprint arXiv:2006.00719*.

Second-Order Sensitivity Analysis for Bilevel Optimization Appendix

A PROOFS

Theorem 2 (Second-Order IFT)**.** *Let $k : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$ be a twice continuously differentiable multivariate function of two variables $z^\star \in \mathbb{R}^m$ and $p \in \mathbb{R}^n$ such that k defines a fixed point for x, i.e. $k(z^\star, p) = 0$ for all values of $p \in \mathbb{R}^n$. Then, the Hessian of (implicitly defined) $z^\star$ w.r.t. p is given by*

$$\begin{aligned} H_p z^\star = -\Big[\big(D_{z^\star} k\big)^{-1} \otimes I\Big]\Big[&H_p k + \big(D_{pz^\star} k\big)\big(D_p z^\star\big) \\ &+ \big(I \otimes \big(D_p z^\star\big)^T\big)\big(D_{z^\star p} k\big) \\ &+ \big(I \otimes \big(D_p z^\star\big)^T\big)\big(H_{z^\star} k\big)\big(D_p z^\star\big)\Big]. \end{aligned} \tag{7}$$

Proof. Differentiation of the implicit expression further requires establishing a convention for matrix expression with vector inputs derivatives. We keep the standard convention of representing the partial Jacobian (with operator D) and scalar function Hessian (with operator H). The missing representations include the Hessian and the 2nd order mixed derivative. Let $f : \mathbb{R}^m \times \mathbb{R}^n \mapsto \mathbb{R}^p$, then we represent Hessian and 2nd order mixed derivatives as

$$\begin{aligned} D_{ab} f(a,b) &= \begin{bmatrix} D_b\big(\nabla_a\big(f_1\big)\big) \\ \vdots \\ D_b\big(\nabla_a\big(f_p\big)\big) \end{bmatrix} && \in \mathbb{R}^{pm \times n} \\ H_a f(a,b) &= \begin{bmatrix} H_a f_1 \\ \vdots \\ H_a f_p \end{bmatrix} && \in \mathbb{R}^{pm \times m}. \end{aligned} \tag{13}$$

We introduce the following derivatives rules

$$D_a\big(A(a)B\big) = \big(I \otimes B^T\big)\big(D_a A(a)\big) \tag{14}$$

$$D_a\big(BA(a)\big) = \big(B \otimes I\big)\big(D_a A(a)\big) \tag{15}$$

where the chain rule applies as

$$D_a\big(A(c(a)\big) = \big(D_c A(c)\big)\big(D_a c(a)\big) \tag{16}$$

This allows to differentiate $D_p k + \big(D_{z^\star} k\big)\big(D_p z^\star\big) = 0$ giving

$$\begin{aligned} &H_p k(z^\star, p) \\ &+ \big(D_{pz^\star} k(z^\star, p)\big)\big(D_p z^\star\big) \\ &+ \big(I \otimes \big(D_p z^\star\big)^T\big)\big(D_{z^\star p} k(z^\star, p)\big) \\ &+ \big(I \otimes \big(D_p z^\star\big)^T\big)\big(H_{z^\star} k(z^\star, p)\big)\big(D_p z^\star\big) \\ &+ \big(\big(D_{z^\star} k(z^\star, p)\big) \otimes I\big)\big(H_p z^\star\big) \\ &= 0 \end{aligned} \tag{17}$$

The implicit Hessian is given by solving Equation (17) for $H_p z^\star$.

$$\begin{aligned} H_p z^\star = -\Big[\big(D_{z^\star} k(z^\star, p)\big)^{-1} \otimes I\Big]\Big[&H_p k(z^\star, p) + \big(D_{pz^\star} k(z^\star, p)\big)\big(D_p z^\star\big) \\ &+ \big(I \otimes \big(D_p z^\star\big)^T\big)\big(D_{z^\star p} k(z^\star, p)\big) \\ &+ \big(I \otimes \big(D_p z^\star\big)^T\big)\big(H_{z^\star} k(z^\star, p)\big)\big(D_p z^\star\big)\Big] \end{aligned} \tag{7}$$

exploiting the identity $\big(A \otimes I\big)^{-1} = A^{-1} \otimes I$. □

Theorem 3 (First-Order Error Bound)**.** *Given the result of IFT applied to an exact solution $D_{z^\star} p = g = -A^{-1}B$, where $A(z^\star, p) = D_{z^\star} k(z^\star, p)$, $B(z^\star, p) = D_p k(z^\star, p)$ and an inexact solution $\widetilde{A}(z, p) = D_z k(z, p)$ (where $k(z, p) \neq 0$) and $\widetilde{B}(z, p) = D_p k(z, p)$. Assume $||z - z^\star|| \leq \delta$, $\Big|\Big|\widetilde{A} - A\Big|\Big|_{op} \leq \gamma\delta$, $\Big|\Big|\widetilde{B} - B\Big|\Big|_F \leq \beta\delta$, $||B||_F \leq R$, $\Big|\Big|\widetilde{A}v\Big|\Big| \geq \alpha_1||v||$, $||Av|| \geq \alpha_2||v||$, then*

$$\Big|\Big|\widetilde{J} - J\Big|\Big|_F \leq \frac{\beta}{\alpha_1}\delta + \frac{\gamma R}{\alpha_1 \alpha_2}\delta \tag{18}$$

Proof.

$$\begin{aligned} -(\widetilde{J} - J) &= \widetilde{A}^{-1}\widetilde{B} - A^{-1}B && (19)\\ &= \widetilde{A}^{-1}\widetilde{B} - \widetilde{A}^{-1}B + \widetilde{A}^{-1}B - A^{-1}B && (20)\\ &= \widetilde{A}^{-1}(\widetilde{B} - B) + (\widetilde{A}^{-1} - A^{-1})B && (21) \end{aligned}$$

which allows to bound the implicit gradient error as

$$\begin{aligned} \Big|\Big|\widetilde{J} - J\Big|\Big|_F &\leq \Big|\Big|\widetilde{A}^{-1}\Big|\Big|_{\text{op}}\Big|\Big|\widetilde{B} - B\Big|\Big|_F + \Big|\Big|\widetilde{A}^{-1} - A^{-1}\Big|\Big|_{\text{op}}||B||_F \\ &\leq \Big|\Big|\widetilde{A}^{-1}\Big|\Big|_{\text{op}}\Big|\Big|\widetilde{B} - B\Big|\Big|_F + \Big|\Big|\widetilde{A}^{-1}\Big|\Big|_{\text{op}}\Big|\Big|\widetilde{A} - A\Big|\Big|_{\text{op}}\big|\big|A^{-1}\big|\big|_{\text{op}}||B||_F \\ &\leq \frac{\beta}{\alpha_1}\delta + \frac{\gamma R}{\alpha_1 \alpha_2}\delta \end{aligned}$$

exploiting the fact that for any invertible matrices M_1, M_2 (where $\widetilde{A}$, A are invertible from $\Big|\Big|\widetilde{A}v\Big|\Big| \geq \alpha_1||v||$, $||Av|| \geq \alpha_2||v||$) $\big(M_1^{-1} - M_2^{-2}\big) = M_1^{-1}\big(M_1 - M_2\big)M_2^{-1}$. □

Theorem 4 (Second-Order Error Bound)**.** *Given the result of IFT applied to an exact solution, for $z^\star \in \mathbb{R}^m$, $H_{z^\star} p = H = -A^{-1}B$, where $A(z^\star, p) = D_{z^\star} k(z^\star, p) \otimes I$, $B(z^\star, p) = H_p k + \big(D_{pz^\star} k\big)\big(D_p z^\star\big) + \big(I \otimes \big(D_p z^\star\big)^T\big)\big(D_{z^\star p} k\big) + \big(I \otimes \big(D_p z^\star\big)^T\big)\big(H_{z^\star} k\big)\big(D_p z^\star\big)$ and an inexact solution $\widetilde{A}(z, p)$, $\widetilde{B}(z, p)$ (where $k(z, p) \neq 0$). Assume $||z - z^\star|| \leq \delta$, $\Big|\Big|\widetilde{A} - A\Big|\Big|_{op} \leq \gamma\delta$, $||H_p k(z, p) - H_p k(z^\star, p)||_F \leq \zeta\delta$, $||D_{zp} k(z, p) - D_{z^\star p} k(z^\star, p)||_F \leq \eta\delta$, $||H_z k(z, p) - H_{z^\star} k(z^\star, p)||_F \leq \nu\delta$, $||D_z p - D_{z^\star} p||_F \leq \kappa_J \delta$ defined in Theorem 3, $||B||_F \leq R_H$, $\Big|\Big|\widetilde{A}v\Big|\Big| \geq \alpha_1||v||$, $||Av|| \geq \alpha_2||v||$, then*

$$\Big|\Big|\widetilde{H} - H\Big|\Big|_F \leq \frac{\zeta + 2\eta\kappa_J + \nu\kappa_J^2}{\alpha_1}\delta + \frac{\gamma R_H}{\alpha_1 \alpha_2}\delta. \tag{22}$$

Proof.

$$\begin{aligned} -(\widetilde{H} - H) &= \Big(\widetilde{A}^{-1} \otimes I\Big)\widetilde{B} - \big(A^{-1} \otimes I\big)B && (23)\\ &= \Big(\widetilde{A}^{-1} \otimes I\Big)\widetilde{B} - \Big(\widetilde{A}^{-1} \otimes I\Big)B + \Big(\widetilde{A}^{-1} \otimes I\Big)B - \big(A^{-1} \otimes I\big)B && (24)\\ &= \Big(\widetilde{A}^{-1} \otimes I\Big)(\widetilde{B} - B) + \Big((\widetilde{A}^{-1} - A^{-1}) \otimes I\Big)B && (25)\\ & && (26) \end{aligned}$$

which allows to bound the implicit Hessian error as

$$\begin{aligned}\left|\left|\widetilde{H}-H\right|\right|_F &\leq \left|\left|\widetilde{A}^{-1}\right|\right|_{\text{op}}\left|\left|\widetilde{B}-B\right|\right|_F + \left|\left|\widetilde{A}^{-1}-A^{-1}\right|\right|_{\text{op}}||B|| \\ &\leq \left|\left|\widetilde{A}^{-1}\right|\right|_{\text{op}}\left|\left|\widetilde{B}-B\right|\right|_F + \left|\left|\widetilde{A}^{-1}\right|\right|_{\text{op}}\left|\left|\widetilde{A}-A\right|\right|_{\text{op}}\left|\left|A^{-1}\right|\right|_{\text{op}}||B||_F \\ &\leq \frac{\zeta+2\eta\kappa_J+\nu\kappa_J^2}{\alpha_1}\delta + \frac{\gamma R_H}{\alpha_1\alpha_2}\delta\end{aligned}$$

exploiting the fact that for any invertible matrices M_1, M_2 (where $\widetilde{A}$, A are invertible from $\left|\left|\widetilde{A}v\right|\right| \geq \alpha_1||v||$, $||Av|| \geq \alpha_2||v||$) $\left(M_1^{-1}-M_2^{-2}\right) = M_1^{-1}\big(M_1-M_2\big)M_2^{-1}$. □

Theorem 5 (Regularized First-Order Error Bound)**.** *Given the result of regularized IFT applied to an exact solution $D_{z^\star}p = g = -A^{-1}B$, where $A(z^\star,p) = D_{z^\star}k(z^\star,p)$, $B(z^\star,p) = D_pk(z^\star,p)$ and an inexact solution $\widetilde{A}(z,p) = D_zk(z,p)+\epsilon I$, $\widetilde{B}(z,p) = D_pk(z,p)$ (where $k(z,p)\neq 0$) . Assume $||z-z^\star|| \leq \delta$, $\left|\left|\widetilde{A}-A\right|\right|_{op} \leq \gamma\delta$, $\left|\left|\widetilde{B}-B\right|\right|_F \leq \beta\delta$, $||B||_F \leq R$, $\left|\left|\widetilde{A}v\right|\right| \geq \alpha_1||v||$ and $v^T\widetilde{A}v \geq 0$, so $\left|\left|\widehat{A}v\right|\right| \geq (\alpha_1+\epsilon)||v||$, $||Av|| \geq \alpha_2||v||$ then*

$$\left|\left|\widehat{J}-J\right|\right|_F \leq \frac{\beta\delta}{\alpha_1+\epsilon} + \frac{R\big(\gamma\delta+\epsilon\big)}{\big(\alpha_1+\epsilon\big)\alpha_2} \tag{27}$$

Proof. We observe that from singular value decomposition

$$\left|\left|\widetilde{A}v\right|\right| \geq \alpha_1||v||, v^T\widetilde{A}v \geq 0 \implies \left|\left|\widetilde{A}+\epsilon I\right|\right|_{\text{op}} \geq \alpha_1+\epsilon \tag{28}$$

which gives

$$\begin{aligned}\left|\left|\widehat{J}-J\right|\right|_F &\leq \left|\left|\widehat{A}^{-1}\right|\right|_{\text{op}}\left|\left|\widetilde{B}-B\right|\right| + \left|\left|\widehat{A}^{-1}-A^{-1}\right|\right|_{\text{op}}||B|| \\ &\leq \left|\left|\widehat{A}^{-1}\right|\right|_{\text{op}}\left|\left|\widetilde{B}-B\right|\right| + \left|\left|\widehat{A}^{-1}\right|\right|_{\text{op}}\left|\left|\widehat{A}-A\right|\right|_{\text{op}}\left|\left|A^{-1}\right|\right|_{\text{op}}||B|| \\ &\leq \left|\left|\widehat{A}^{-1}\right|\right|_{\text{op}}\left|\left|\widetilde{B}-B\right|\right| + \left|\left|\widehat{A}^{-1}\right|\right|_{\text{op}}\left|\left|\widetilde{A}+\epsilon I-A\right|\right|_{\text{op}}\left|\left|A^{-1}\right|\right|_{\text{op}}||B|| \\ &\leq \left|\left|\widehat{A}^{-1}\right|\right|_{\text{op}}\left|\left|\widetilde{B}-B\right|\right| + \left|\left|\widehat{A}^{-1}\right|\right|_{\text{op}}\left(\left|\left|\widetilde{A}-A\right|\right|_{\text{op}}+||\epsilon I||_{\text{op}}\right)\left|\left|A^{-1}\right|\right|_{\text{op}}||B|| \\ &\leq \frac{\beta}{\alpha_1+\epsilon}\delta + \frac{R}{\big(\alpha_1+\epsilon\big)\alpha_2}\left(\gamma\delta+\epsilon\right)\end{aligned}$$

which gives

$$\left|\left|\widehat{J}-J\right|\right|_F \leq \frac{\beta\delta}{\alpha_1+\epsilon} + \frac{R\big(\gamma\delta+\epsilon\big)}{\big(\alpha_1+\epsilon\big)\alpha_2}. \tag{29}$$

□

B COMPUTATIONAL COMPLEXITY

Evaluating $A \otimes I$ Efficiently The product $A \otimes I$ features in the Hessian expression

$$H_p z^\star = \left[\left(D_{z^\star} k \right)^{-1} \otimes I \right] [\dots]$$

Given the expression $M = \big(A \otimes I\big)C$, let $A \in \mathbb{R}^{m \times n}$, $C \in \mathbb{R}^{(np) \times r}$, such that

$$C = \begin{bmatrix} C_1 \in \mathbb{R}^{p \times r} \\ \vdots \\ C_n \in \mathbb{R}^{p \times r} \end{bmatrix}$$

Let $M = \big(A \otimes I\big)C$ and define $\widetilde{C} \in \mathbb{R}^{n \times p \times r}$ s.t. $\widetilde{C}_{i**} \in \mathbb{R}^{p \times r} \;\; \forall i \in [1..n]$, $\widetilde{M} \in \mathbb{R}^{m \times p \times r}$ s.t. $\widetilde{M}_{j**} \in \mathbb{R}^{p \times r} \;\; \forall j \in [1..m]$—a 3-d representation of C and M where the stacked matrices in C, M are concatenated along a third, first, dimension in $\widetilde{C}$, $\widetilde{M}$.

The operation can now be defined formally in Einstein summation notation as

$$\big(\widetilde{M}\big)_{ijk} = \big(A\big)_{il} \big(\widetilde{C}\big)_{ljk}$$

Intuitively, the operation $M = \big(A \otimes I\big)$ corresponds to a matrix multiplication performed for every vector in C built from n elements, one from each C_i.

Evaluating $I \otimes B$ Efficiently The product $I \otimes B$ features in the Hessian expression

$$H_p z^\star = [\dots] \left[\cdots + \left(I \otimes \left(D_p z^\star \right)^T \right) \left(D_{z^\star p} k \right) + \dots \right]$$

Given the expression $M = \big(I \otimes B\big)C$, let $A \in \mathbb{R}^{m \times n}$, $C \in \mathbb{R}^{(pn) \times r}$, such that

$$C = \begin{bmatrix} C_1 \in \mathbb{R}^{n \times r} \\ \vdots \\ C_p \in \mathbb{R}^{n \times r} \end{bmatrix}$$

Let $M = \big(I \otimes B\big)C$ and define $\widetilde{C} \in \mathbb{R}^{p \times n \times r}$ s.t. $\widetilde{C}_{i**} \in \mathbb{R}^{n \times r} \;\; \forall i \in [1..n]$, $\widetilde{M} \in \mathbb{R}^{p \times m \times r}$ s.t. $\widetilde{M}_{j**} \in \mathbb{R}^{m \times r} \;\; \forall j \in [1..p]$—a 3-d representation of C and M where the stacked matrices in C, M are concatenated along a third, first, dimension in $\widetilde{C}$, $\widetilde{M}$.

The operation can now be defined formally in Einstein summation notation as

$$\big(\widetilde{M}\big)_{ijk} = \big(A\big)_{jl} \big(\widetilde{C}\big)_{ilk}$$

Intuitively, the operation $M = \big(I \otimes B\big)$ corresponds to a matrix multiplication performed by broadcasting B and performing matrix multiplication of B with every p element of C, so that $BC_i \;\; \forall i \in [1..p]$—these products are then stacked together to form M.

C ADDITIONAL INSIGHTS

Convexity-in-parameter Global convexity-in-parameter is obtained if the second-order order derivative of the upper loss function w.r.t. to the parameter is globally positive semi-definite (PSD) while both the objective functions are globally twice continuously differentiable. Recalling the full derivative expressions

$$\begin{aligned} Hf_U(z^\star, p) = H_p f_U + \big(D_p z^\star\big)^T \big(H_{z^\star} f_U\big) \big(D_p z^\star\big) + \\ + \big(\big(D_{z^\star} f_U\big) \otimes I\big) H_p z^\star \end{aligned}$$

we conclude that it is not possible to guarantee the global PSD property for this Hessian in general, because of the reduction $\big(\big(D_{z^\star} f_U\big) \otimes I\big) H_p z^\star$ as outlined in Section B.[8] However, in the special case when $H_p z^\star = 0$ and the problem is globally twice differentiable, global convexity is obtained, e.g., unconstrained quadratic programs with linear parameterizations.

[8] A stack of Hessian matrices are reduced by a weighted summation with unknown sign weights.