Московский физико-технический институт
(национальный исследовательский университет)
Физтех-школа прикладной математики и информатики
Кафедра интеллектуальных систем

Яковлев Константин Дмитриевич

Обобщенная жадная градиентная оптимизация гиперпараметров

03.04.01 — Прикладные математика и физика

Выпускная квалификационная работа магистра

Научный руководитель:
к.ф.-м.н. О. Ю. Бахтеев

Москва — 2024

Abstract

Bilevel Optimization (BLO) is a widely-used approach that has numerous applications, including hyperparameter optimization, meta-learning. However, existing gradient-based method suffer from the following issues. Reverse-mode differentiation suffers from high memory requirements, while the methods based on the implicit function theorem require the convergence of the inner optimization. The approximations that consider a truncated inner optimization trajectory suffer from a short horizon bias. In this paper, we propose a novel approximation for hypergradient computation that sidesteps this difficulties. Specifically, we accumulate the short-horizon approximations from each step of the inner optimization trajectory. We also show that, under certain conditions, the proposed hypergradient is a sufficient descent direction. Experimental results on a few-shot meta-learning task support our findings.

Содержание

1 Introduction

Bilevel optimization has become an essential component of machine learning, which includes Neural Architecture Search [1, 2, 3], Hyperparameter Optimization [4], and Meta-Learning [5, 6, 7]. In the hierarchical optimization framework, the outer-level objective is aimed to be minimize given the optimality in the inner level. Solving the bilevel problem is challenging due to the intricate dependency of the optimal inner parameters given the outer parameters.

Naive approaches such as random search and grid search [8] become impractical with the growing number of hyperparameters to be optimized due to the curse of dimensionality. Another approach that has proven effective in low-dimensional setting is Bayesian Optimization [9]. However, its extension to high-dimensional setting is challenging [10].

In the current work we develop a novel gradient-based algorithm [11]. The challenge is that the exact hypergradient calculation is computationally demanding [12]. Specifically, Forward-Mode differentiation (FMD) is memory demanding, since it increases linearly with the number of hyperparameters. This limits the application of the method for large-scale problems with millions of hyperparameters, such as meta-learning. By contrast, Revers-Mode Differentiation (RMD) perfectly scales to problems with millions of hyperparameters, but it requires the full inner optimization trajectory of model parameters to be saved, which is computationally costly. Moreover, RMD suffers from gradient vanishing or explosion [13], which leads to training instability. Truncation of the optimization trajectory was proposed to alleviate high memory consumption [14] while calculating an approximate hypergradient. However, this approach suffers from short horizon bias [15]. Following [16], we define greediness as finding the optimal hyperparameters on a local scale, rather than on a global scale.

Alternatively, an implicit differentiation may be used to compute the hypergradient [17, 18, 19]. This approach mitigates the need for unrolling, but it heavily relies on Implicit Function Theorem, which requires the convergence of the inner optimization [20, 21]. The challenge of this family of methods is computing inverse Hessian-vector

Таблица 1: Comparison of gradient-based methods for hyperparameter optimization. Here, P represents the number of model parameters, and H represents the number of hyperparameters. Additionally, K represents the number of terms in the Neumann approximation.

	RMD [12]	**FMD** [12]	**IFT** [17]	**T1 – T2** [18]	**Ours**
Long Horizon	Yes	Yes	Yes	No	Yes
Scalable to large amount of hyperparameters	Yes	No	Yes	Yes	Yes
Space Complexity	$O(PT)$	$O(PH)$	$O(P+H)$	$O(P+H)$	$O(P+H)$
Time complexity	$O(T)$	$O(HT)$	$O(K)$	$O(1)$	$O(T)$
No inner optimality	Yes	Yes	No	Yes	Yes

product. This computation may be approximated with Neumann series [17] or conjugate gradients [19].

In this paper, we propose an alternative approach to hypergradient computation. We generalize the method from [18]. Namely, the proposed approach resolves the following issues simultaneously: short horizon bias, high memory requirements, applicability to large-scale problems with millions of hyperparameters, and independence of inner optimization convergence. Overall, our contributions are as follows:

1. we introduce a procedure that aggregates the greedy gradients calculated at each iteration of the inner objective, which satisfies the requirements above.

2. We provide a theoretical analysis of the proposed approach. Under some assumptions, a sufficient descent condition holds.

3. We empirically prove the effectiveness of the proposed approach on a Meta-Learning task.

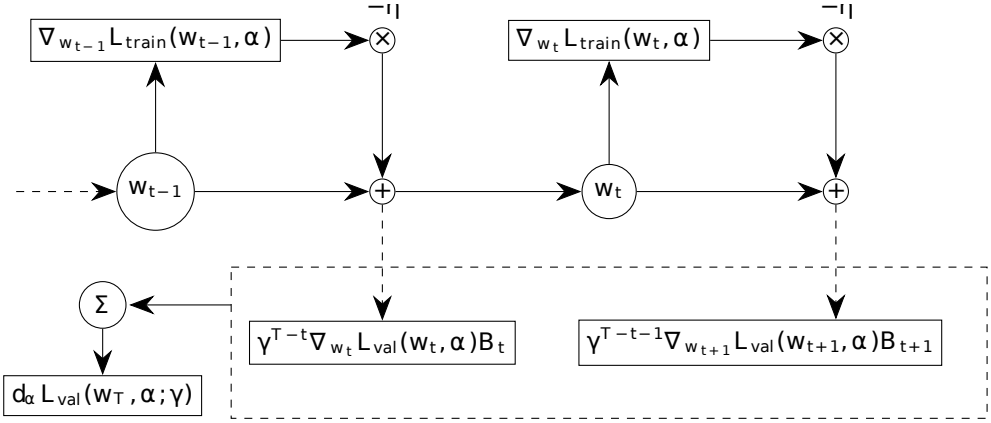

Рис. 1: The schematic illustration of the proposed approach. In general, the approximate hypergradient is calculated as a weighted sum of the locally optimal greedy gradients calculated at each inner optimization step.

2 Related Work

Gradient-Based Hyperaprameter Optimization. Differentiation through optimization [22] was successfully applied to hyperparameter optimization at a large-scale [23]. The unrolled differentiation could be categorized into Forward-Mode and Reverse-Mode differentiation [12]. The former one suits best for the cases when a handful of hyperparameters is needed to be optimized [16], for instance, learning rate and weight dacay. The latter is suitable for the setup with millions of hyperparameters while sacrificing the memory consumption when the number of inner optimization steps increases, except for the cases when SGD with momentum is used [23]. Additionally, truncated unrolled differentiation [14] introduces a trade-off between computational complexity and hypergradient accuracy. However, computations done on truncated trajectories suffer from short horizon bias [15].

Alternatively, impicit differentiation, inspired by the Implicit Function Theorem (IFT), is used to compute the hypergradient [11, 17, 18, 19]. In [11] an exact inverse Hessian is computed, which is computationally intractable in huge-scale scenario with millions of model parameters. To sidestep this issue, an approximation is needed. Specifically, the Neumann series approximation [17], conjugate gradients [19], GMRES [21] for solving linear systems, Nyström method [24], and Broyden's method [25]. The major limitation is that the near-optimality of the inner optimization is crucial for accurate approximation of the true hypergradient [20, 21]. Moreover, the method is inapplicable to tackling the optimizer hyperaprameters such as learning rate.

We summarize the comparison of described approaches in Table 1.

Meta-Learning. Another fundamental application of bilevel optimization is meta-learning [26] (or learning to learn). It aims to train a model that generalizes well over the distribution of tasks [27]. In the context of gradient-based model-agnostic meta-learning [7], the task is to learn an initialization of model parameters such that gradient-based fine-tuning shows good generalization. MAML optimization successfully inherts the methods for hypergradient computation. Specifically, [28] successfully employed [18], [29] used implicit differentiation with conjugate gradient

algorithm.

3 Background

In this section we introduce a derivation of an exact hypergradient computation.

3.1 Hypergradient computation

Given a vector of model parameters $\mathbf{w} \in \mathbb{R}^P$ and a vector of hyperparameters $\boldsymbol{\alpha} \in \mathbb{R}^H$. The dynamic of model parameters $\{\mathbf{w}_t\}_{t=0}^T$ for some $T \in \mathbb{N}$ and some $\boldsymbol{\alpha}$ is defined as follows $\mathbf{w}_{t+1} = \boldsymbol{\Phi}(\mathbf{w}_t, \boldsymbol{\alpha})$, where $\boldsymbol{\Phi}(.,.)$ is a smooth mapping. For instance, a vanilla gradient descent with stepsize $\eta > 0$ could be written as $\boldsymbol{\Phi}(\mathbf{w}_t, \boldsymbol{\alpha}) = \mathbf{w}_t - \eta \nabla_{\mathbf{w}} \mathcal{L}_{\text{train}}(\mathbf{w}_t, \boldsymbol{\alpha})$, where $\mathcal{L}_{\text{train}}$ is a training loss function. Given also a differentiable validation loss function $\mathcal{L}_{\text{val}}(\mathbf{w}, \boldsymbol{\alpha})$. Under the notations above we formulate a hyperparameter optimization problem as follows:

$$\boldsymbol{\alpha}^* - \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^H} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha}), \tag{1}$$

$$\text{s.t.} \quad \mathbf{w}_t = \boldsymbol{\Phi}(\mathbf{w}_{t-1}, \boldsymbol{\alpha}), \quad t \in \overline{1, T}. \tag{2}$$

Now the goal is to derive a hypergdadient $d_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha})$, viewing $\mathbf{w}_T$ as a function of $\boldsymbol{\alpha}$:

$$d_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha}) = \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha}) + \nabla_{\mathbf{w}_T} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha}) \frac{d\mathbf{w}_T}{d\boldsymbol{\alpha}}. \tag{3}$$

Here $\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha})$ is a row-vector. The chain rule suggests that $d\mathbf{w}_T / d\boldsymbol{\alpha}$ is computed in the following way [12]:

$$\frac{d\mathbf{w}_T}{d\boldsymbol{\alpha}} = \sum_{t=1}^{T} \left(\prod_{k=t+1}^{T} \mathbf{A}_k \right) \mathbf{B}_t, \quad \mathbf{A}_k = \frac{\partial \boldsymbol{\Phi}(\mathbf{w}_{k-1}, \boldsymbol{\alpha})}{\partial \mathbf{w}_{k-1}}, \quad \mathbf{B}_t = \frac{\partial \boldsymbol{\Phi}(\mathbf{w}_{t-1}, \boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}}. \tag{4}$$

Therefore, the hypergradient is calculated as follows:

$$d_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha}) = \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha}) + \sum_{t=1}^{T} \nabla_{\mathbf{w}_T} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha}) \left(\prod_{k=t+1}^{T} \mathbf{A}_k \right) \mathbf{B}_t. \tag{5}$$

The computation of (4) could be implemented with a Reverse-Mode Differentiation (RMD) or Forward-Mode Differentiation (FMD) [12]. However, the aforementioned method is computationally expensive in terms of either latency (FMD) or memory (RMD). Note that RMD may not need to store the trajectory $\mathbf{w}_1, \ldots, \mathbf{w}_T$ in case of SGD with momentum. However, this would require $2T - 1$ Jacobian-vector products (JVPs), which is computationally demanding. So, we develop the method that performs only T JVPs for the hypergradient computation.

4 The Method

4.1 Hypergradient approximation

In this section we introduce a computationally efficient approximation to (5). Specifically, consider the t-th step of the inner optimization. The challenge is that the computation of $\prod_{k=t+1}^{T} \mathbf{A}_k$ requires the tail of the trajectory $\mathbf{w}_t, \dots, \mathbf{w}_T$. To this end, we introduce an approximation of the product with γ^{T-t}, where $\gamma \in (0, 1]$. We motivate the choice of γ by the fact that $(1 - \eta L)\mathbf{I} \preceq \mathbf{A}_k \preceq \mathbf{I}$ if $\mathcal{L}_{\text{train}}(., \boldsymbol{\alpha})$ is L-smooth and convex for any $\boldsymbol{\alpha} \in \mathbb{R}^H$. Indeed, if we assume that $\boldsymbol{\Phi}(., .)$ is a vanilla gradient descent, then $\mathbf{A}_k = \mathbf{I} - \eta \nabla^2_{\mathbf{w}_{k-1}} \mathcal{L}_{\text{train}}(\mathbf{w}_{k-1}, \boldsymbol{\alpha})$. Due to the convexity and L-smoothness of $\mathcal{L}_{\text{train}}(., \boldsymbol{\alpha})$ we conclude that $\mathbf{0} \preceq \nabla^2_{\mathbf{w}_{k-1}} \mathcal{L}_{\text{train}}(\mathbf{w}_{k-1}, \boldsymbol{\alpha}) \preceq L\mathbf{I}$. So, choosing the step size $\eta \leq L^{-1}$, we conclude that the spectrum of $\mathbf{A}_k$ is bounded between 0 and 1 for any choice of $\boldsymbol{\alpha}$ and k. Additionally, we replace the gradient of the validation loss $\nabla_{\mathbf{w}_T} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha})$ with the gradient from the current iteration $\nabla_{\mathbf{w}_t} \mathcal{L}_{\text{val}}(\mathbf{w}_t, \boldsymbol{\alpha})$ due to the same reason. Write down the proposed approximation:

$$\hat{d}_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha}; \gamma) = \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha}) + \sum_{t=1}^{T} \gamma^{T-t} \nabla_{\mathbf{w}_t} \mathcal{L}_{\text{val}}(\mathbf{w}_t, \boldsymbol{\alpha}) \mathbf{B}_t. \tag{6}$$

Note that the intuition from (6) was previously used in [30]. However, it was used as an intermediate step in the reasoning. Moreover, the approximation of the gradient of the validation loss function w.r.t. model parameters was not considered. Figure **??** shows a schematic overview of the propsed approach.

4.2 Generalization of $T1 - T2$

Note that the proposed hypergradient computation (6) is a generalization of $T1 - T2$ hypergradient [18] when γ tends to zero. Below we formulate a formal statement.

Proposition 4.1. *Let $\hat{d}_{\boldsymbol{\alpha}}(\mathbf{w}_T, \boldsymbol{\alpha}; \gamma)$ be the hypergradient defined in* (6)*. Then, the*

following holds:

$$\lim_{\gamma \to 0^+} \hat{d}_{\boldsymbol{\alpha}}(\mathbf{w}_T, \boldsymbol{\alpha}; \gamma) = \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha}) + \nabla_{\mathbf{w}_T} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha}) \mathbf{B}_T. \tag{7}$$

Доказательство. First, using the definition of $\hat{d}_{\boldsymbol{\alpha}}(\mathbf{w}_T, \boldsymbol{\alpha}; \gamma)$ from (6), we conclude that:

$$\hat{d}_{\boldsymbol{\alpha}}(\mathbf{w}_T, \boldsymbol{\alpha}; \gamma) = \nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha}) + \nabla_{\mathbf{w}_T} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha}) \mathbf{B}_T + \gamma \sum_{t=1}^{T-1} \gamma^{T-t-1} \nabla_{\mathbf{w}_t} \mathcal{L}_{\text{val}}(\mathbf{w}_t, \boldsymbol{\alpha}) \mathbf{B}_t.$$

Second, note that the last term tends to zero:

$$\lim_{\gamma \to 0^+} \gamma \sum_{t=1}^{T-1} \gamma^{T-t-1} \nabla_{\mathbf{w}_t} \mathcal{L}_{\text{val}}(\mathbf{w}_t, \boldsymbol{\alpha}) \mathbf{B}_t = \mathbf{0}.$$

The combination of the above two steps completes the proof. □

Here the right hand side of (7) is the hypergradient of in $T1 - T2$ [18]. The result given in Proposition 4.1 suggest that $T1 - T2$ hypergradient is a special case of the proposed one. Additionally, it could be clearly seen that the proposed hypergradient computation is conditioned on the whole trajectory of model parameters. We argue that this approach does not suffer from a short-horizon bias problem [15].

4.3 Descent Direction Analysis

Here we discuss the quality of the proposed hypergradient approximation (6). We show that the sufficient descent condition holds under some assumptions. Inspired by [14, 31], we first formulate a standard set of assumptions.

Assumption 4.2. *Suppose that the following assumptions on the functions* $\mathcal{L}_{\text{train}}(.,.)$, $\mathcal{L}_{\text{val}}(.,.)$, *and the optimization operator* $\boldsymbol{\Phi}(.,.)$ *are satisfied:*

1. $\mathcal{L}_{\text{val}}(., \boldsymbol{\alpha})$ *is* L*-smooth and* μ*-strongly convex for any* $\boldsymbol{\alpha}$.
2. $\frac{\partial \boldsymbol{\Phi}(.,\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}}$ *is* C_B*-Lipschitz for any* $\boldsymbol{\alpha}$.

3. $\|\frac{\partial \mathbf{\Phi}(\mathbf{w},\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}}\|_{op} \leq B$ *for any pair* $(\mathbf{w}, \boldsymbol{\alpha})$ *for some* $B \geq 0$.

4. $\mathbf{w}$ *belongs to a bounded convex set with diameter* $D < \infty$.

5. $\mathbf{\Phi}(\mathbf{w}, \boldsymbol{\alpha}) = \mathbf{w} - \eta \nabla_{\mathbf{w}} \mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\alpha})$ *for some* $\eta \geq 0$.

Second, we formulate and justify specific assumptions.

Assumption 4.3. *Suppose that the following holds for* $\mathcal{L}_{\text{train}}(.,.)$ *and* $\mathcal{L}_{\text{val}}(.,.)$*:*

1. $\nabla^2_{\mathbf{w}} \mathcal{L}_{\text{train}}(., \boldsymbol{\alpha}) = \mathbf{I}$ *for any* $\boldsymbol{\alpha}$*. Note that this assumption does not hold in practice. However, [18] argues that batch normalization [32] forces the Hessian to be close to the identity matrix.*

2. $\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w}, \boldsymbol{\alpha}) = \mathbf{0}$ *for any* $\mathbf{w}$*. This assumption is typical for hyperparameter optimization and data hypercleaning [12].*

3. $\mathbf{B}_t \mathbf{B}_t^\top \succeq \kappa \mathbf{I}$ *for some* $\kappa > 0$*. We note that the assumption that* $\mathbf{B}_t$ *is a full-rank matrix was used in [14]. However, we impose more strict assumption to simplify the proofs.*

4. *Define* $\mathbf{w}_\infty := \arg\min_{\mathbf{w}} \mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\alpha})$, $\mathbf{w}_2^* := \arg\min_{\mathbf{w}} \mathcal{L}_{\text{val}}(\mathbf{w}, \boldsymbol{\alpha})$. *Assume that* $\|\mathbf{w}_\infty - \mathbf{w}_2^*\| \geq 2De^{-\mu\eta T} + \delta$*, for some* $\delta > 0$*. Also assume that* $\nabla_{\mathbf{w}_2^*} \mathcal{L}_{\text{val}}(\mathbf{w}_2^*, \boldsymbol{\alpha}) = \mathbf{0}$ *for any* $\boldsymbol{\alpha}$*. Intuitively, this requirements asserts that an overfitting takes place, and the minimum is reached in the interior of the feasible set.*

Lemma 4.4. *([14]) In the assumptions above 4.2, 4.3, the sequence* $\{\mathbf{w}_t\}_{t \geq 0}$ *satisfies:*

$$\|\mathbf{w}_t - \mathbf{w}_\infty\|_2 \leq \|\mathbf{w}_0 - \mathbf{w}_\infty\|_2 e^{-\eta t}. \tag{8}$$

Lemma 4.5. *Let the assumptions 4.2, 4.3 hold. Then the following is true:*

$$\|\nabla_{\mathbf{w}_T} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha})\|_2 \geq \mu\delta. \tag{9}$$

Доказательство. First, use the Polyak-Lojasiewicz condition, since $\mathcal{L}_{\text{val}}(.,.)$ is μ-strongly convex in the first argument due to 4.2. Second, use the strong convexity of $\mathcal{L}_{\text{val}}(., \boldsymbol{\alpha})$ according to 4.2. Third, use Lemma 4.4 for $\mathbf{w}_T$, and finally the overfitting condition from 4.3:

$$\begin{aligned}
\|\nabla_{\mathbf{w}_T}\mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha})\|_2^2 &\overset{4.2(1)}{\geq} 2\mu(\mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha}) - \mathcal{L}_{\text{val}}(\mathbf{w}_2^*, \boldsymbol{\alpha})) \\
&\overset{4.2(1)}{\geq} \mu^2\|\mathbf{w}_T - \mathbf{w}_2^*\|^2 \\
&\geq \mu^2(\|\mathbf{w}_T - \mathbf{w}_\infty\|_2^2 + \|\mathbf{w}_2^* - \mathbf{w}_\infty\|_2^2 - 2\|\mathbf{w}_T - \mathbf{w}_\infty\|_2 \cdot \|\mathbf{w}_2^* - \mathbf{w}_\infty\|_2) \\
&\overset{4.4}{\geq} \mu^2(\|\mathbf{w}_2^* - \mathbf{w}_\infty\|_2 - 2De^{-\mu\eta T})\|\mathbf{w}_2^* - \mathbf{w}_\infty\|_2 \\
&\overset{4.3(4)}{\geq} \mu^2\delta^2.
\end{aligned}$$

□

The following theorem guarantees that the proposed hypergradient is a sufficient descent direction.

Theorem 4.6. *Suppose that $\gamma = 1 - \eta \in (0, 1)$. Then, under the assumptions above 4.2, 4.3, there exists a sufficiently large T and a universal constant $c > 0$ such that:*

$$d_{\boldsymbol{\alpha}}\mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha})\hat{d}_{\boldsymbol{\alpha}}\mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha}; \gamma)^\top \geq c\|d_{\boldsymbol{\alpha}}\mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha})\|_2^2.$$

Доказательство. Define $\mathbf{g}_j := \nabla_{\mathbf{w}_j}\mathcal{L}_{\text{val}}(\mathbf{w}_j, \boldsymbol{\alpha})$ for $j \in \{1, \ldots, T\}$. Write down the dot product taking into account that $\prod_{k=t+1}^{T} \mathbf{A}_k = (1-\eta)^{T-t}$ according to 4.3(1):

$$\begin{aligned}
d_{\boldsymbol{\alpha}}\mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha})\hat{d}_{\boldsymbol{\alpha}}\mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha}; \gamma)^\top &= \sum_{j=1}^{T}\sum_{t=1}^{T}(1-\eta)^{2T-t-j}\nabla_{\mathbf{w}_T}\mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha})\mathbf{B}_t\mathbf{B}_j^\top\nabla_{\mathbf{w}_j}\mathcal{L}_{\text{val}}(\mathbf{v} \\
&= \sum_{j=1}^{T}\sum_{t=1}^{T}(1-\eta)^{2T-j-t}\mathbf{g}_T\mathbf{B}_t\mathbf{B}_j^\top\mathbf{g}_j.
\end{aligned}$$

Now estimate each term from below

$$\begin{aligned}
\mathbf{g}_T\mathbf{B}_t\mathbf{B}_j^\top\mathbf{g}_j &= \mathbf{g}_T\mathbf{B}_t\mathbf{B}_t^\top\mathbf{g}_j + \mathbf{g}_T\mathbf{B}_t(\mathbf{B}_j-\mathbf{B}_t)^\top\mathbf{g}_j \\
&\overset{4.2(2)}{\geq} \mathbf{g}_T\mathbf{B}_t\mathbf{B}_t^\top\mathbf{g}_j - C_B\|\mathbf{w}_j-\mathbf{w}_t\|_2\cdot\|\mathbf{g}_j\|_2\cdot\|\mathbf{g}_T\|_2\cdot\|\mathbf{B}_t\|_{\text{op}} \\
&\overset{4.3(4),4.2(3)}{\geq} \mathbf{g}_T\mathbf{B}_t\mathbf{B}_t^\top\mathbf{g}_j - C_BB\|\mathbf{w}_j-\mathbf{w}_t\|_2\cdot\|\mathbf{g}_j-\nabla_{\mathbf{w}_2^*}\mathcal{L}_{\text{val}}(\mathbf{w}_2^*,\boldsymbol{\alpha})\|_2\cdot\|\mathbf{g}_T\|_2 \\
&\overset{4.2(1)}{\geq} \mathbf{g}_T\mathbf{B}_t\mathbf{B}_t^\top\mathbf{g}_j - C_BB\|\mathbf{w}_j-\mathbf{w}_t\|_2\cdot L\|\mathbf{w}_j-\mathbf{w}_2^*\|_2\cdot\|\mathbf{g}_T\|_2 \\
&\geq \mathbf{g}_T\mathbf{B}_t\mathbf{B}_t^\top\mathbf{g}_j - C_BBLD\|\mathbf{w}_j-\mathbf{w}_\infty+\mathbf{w}_\infty-\mathbf{w}_t\|_2\|\mathbf{g}_T\|_2 \\
&\overset{(8)}{\geq} \mathbf{g}_T\mathbf{B}_t\mathbf{B}_t^\top\mathbf{g}_j - C_BBLD(\|\mathbf{w}_0-\mathbf{w}_\infty\|_2e^{-\eta t}+\|\mathbf{w}_0-\mathbf{w}_\infty\|_2e^{-\eta j})\|\mathbf{g}_T\|_2 \\
&\overset{4.2(4)}{\geq} \mathbf{g}_T\mathbf{B}_t\mathbf{B}_t^\top\mathbf{g}_j - C_BBLD^2(e^{-\eta t}+e^{-\eta j})\|\mathbf{g}_T\|_2
\end{aligned}$$

Now bound $\mathbf{g}_T\mathbf{B}_t\mathbf{B}_t^\top\mathbf{g}_j$ from below:

$$\begin{aligned}
\mathbf{g}_T\mathbf{B}_t\mathbf{B}_t^\top\mathbf{g}_j &= \mathbf{g}_T\mathbf{B}_t\mathbf{B}_t^\top\mathbf{g}_T + \mathbf{g}_T\mathbf{B}_t\mathbf{B}_t^\top(\mathbf{g}_j-\mathbf{g}_T) \\
&\overset{4.2(1)(3)}{\geq} \kappa\|\mathbf{g}_T\|_2^2 - L\|\mathbf{g}_T\|_2B^2\|\mathbf{w}_j-\mathbf{w}_T\|_2 \\
&\overset{(8)}{\geq} \kappa\|\mathbf{g}_T\|_2^2 - L\|\mathbf{g}_T\|_2B^2\|\mathbf{w}_0-\mathbf{w}_\infty\|_2(e^{-\eta T}+e^{-\eta j}) \\
&\overset{4.2(4)}{\geq} \kappa\|\mathbf{g}_T\|_2^2 - LDB^2\|\mathbf{g}_T\|_2(e^{-\eta T}+e^{-\eta j}).
\end{aligned}$$

Combining together the above bounds, we have:

$$\begin{aligned}
&\sum_{j=1}^T\sum_{t=1}^T(1-\eta)^{2T-j-t}\mathbf{g}_T\mathbf{B}_t\mathbf{B}_j^\top\mathbf{g}_j \geq \\
&\kappa T^2\|\mathbf{g}_T\|_2^2 - C_BBLD^2\|\mathbf{g}_T\|_2\sum_{j=1}^T\sum_{t=1}^T[e^{-\eta t}+e^{-\eta j}] - LDB^2\|\mathbf{g}_T\|_2(T^2e^{-\eta T}+T\sum_{j=1}^T e^{-\eta j}) \geq \\
&\kappa T^2\|\mathbf{g}_T\|_2^2 - 2C_BBLD^2\|\mathbf{g}_T\|_2T(e^\eta-1)^{-1} - LDB^2\|\mathbf{g}_T\|_2(T^2e^{-\eta T}+T\eta^{-1}) \geq \\
&\kappa T^2\|\mathbf{g}_T\|_2^2 - 2C_BBLD^2\|\mathbf{g}_T\|_2T(e^\eta-1)^{-1} - LDB^2\|\mathbf{g}_T\|_2(T^2e^{-\eta T}+T(e^\eta-1)^{-1}).
\end{aligned}$$

Using Lemma 4.5 we make the following statement. Since the first term of the bound is $\Theta(T^2)$ and the second and the third are $\Theta(T)$, then there exists sufficiently large T and a universal constant c such that the expression is bounded from below with $c\|\mathbf{g}_T\|_2^2$ for $\|\mathbf{g}_T\|_2 \geq \mu\delta$. □

5 Experiments

In this section we present numerical experiments that validate the effectiveness and efficiency of the proposed approach. Upon acceptance, we will make the source codes available.

5.1 Baselines

For comparison, we consider the following list of baselines that are efficient in terms of space and latency:

- **T1 − T2** [18]. The method performs an unrolled differentiation using only the last step of inner optimization, so it performs a JVP.
- **IFT** [17]. The method combines the implicit function theorem (IFT) with efficient approximations of the inverse Hessian. The number of JVPs is controlled by the number of terms taken from the Neumann series.
- **FO**. The method uses only the first-order gradient from (5), namely $\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\text{val}}(\mathbf{w}_T, \boldsymbol{\alpha})$. Note that it is not applicable for tasks for which the outer objective does not depend explicitly on the vector of hyperparameters $\boldsymbol{\alpha}$.

5.2 Gradient-based Meta-Learning

We consider gradient-base Meta-Learning task for few-shot image classification task [7] in a K-shot m-way setting. As for the model, we consider a 3-layer convolutional network with 32 channels. Inspired by [33, 34], we treat the logits head as a model parameters and the backbone of the convolutional network as hyperparameters. We conduct the experiment using CIFAR100 dataset [35] that contains 100 classes, which are ranomly splited into 50 for meta-training and 50 for meta-validation. The training and validation splits for each task consist of K samples for each class.

Method	**#JVPs**	**3-way, 10-shot**	**4-way, 10-shot**	**5-way, 10-shot**
FO	0	43.48 ± 1.64	34.15 ± 1.28	28.59 ± 1.0
$T1 - T2$	1	42.96 ± 1.89	33.95 ± 1.53	27.59 ± 0.99
IFT	11	40.14 ± 2.26	33.23 ± 0.99	27.20 ± 1.12
Ours ($\gamma = 0.99$)	10	$\mathbf{46.10 \pm 1.95}$	$\mathbf{36.94 \pm 2.55}$	$\mathbf{29.79 \pm 1.33}$

Таблица 2: Few-shot accuracy of the meta-learning task.

The inner optimization is done using SGD with a learning rate of 10^{-1} and momentum 0.9, while the outer problem is optimized with Adam with a learning rate of 10^{-3}. The number of outer steps is set to 200 and set $T = 10$. Additionally, meta-batchsize is set to 10. We tune γ for the proposed approach within the set $\{0.9,\, 0.99, 0.999\}$ and select the best-performing value for each task using the meta-validation split. Interestingly, $\gamma = 0.99$ performs remarkably well irrespective of the task.

The accuracy on meta-validation split is presented in Table 2 for different few-shot scenarios, along with the number of JVPs. We report the mean and a 95% confidence interval based on 20 trials using different random seeds . It could be clearly seen that the proposed approach shows substantial improvements over the baselines in terms of accuracy on the meta-validation split.

6 Conclusion

The paper presents an approximation of the true hypergradient for gradient-based bilevel optimization that avoids the high memory cost and short horizon bias. Additionally, the method does not require the assumption of convergence to an optimal solution for the inner optimization. The proposed method exploits an aggregation of greedy gradients calculated at each step of the inner trajectory. Our theoretical findings suggest that the approximation satisfies the sufficient descent condition. Empirically, the introduced method outperforms the baselines in terms of validation accuracy, having comparable computational costs. One promising direction for future research is to investigate more accurate Hessian approximations.

Список литературы

[1] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[2] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pages 4095–4104. PMLR, 2018.

[3] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[4] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[5] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

[6] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[8] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[9] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[10] Xilu Wang, Yaochu Jin, Sebastian Schmitt, and Markus Olhofer. Recent advances in bayesian optimization. *ACM Computing Surveys*, 55(13s):1–36, 2023.

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

[12] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization. In *International Conference on Machine Learning*, pages 1165–1173. PMLR, 2017.

[13] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. In *International conference on learning representations*, 2018.

[14] Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots. Truncated back-propagation for bilevel optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1723–1732. PMLR, 2019.

[15] Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger Grosse. Understanding short-horizon bias in stochastic meta-optimization. *arXiv preprint arXiv:1803.02021*, 2018.

[16] Paul Micaelli and Amos Storkey. Non-greedy gradient-based hyperparameter optimization over long horizons. 2020.

[17] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International conference on artificial intelligence and statistics*, pages 1540–1552. PMLR, 2020.

[18] Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko. Scalable gradient-based tuning of continuous regularization hyperparameters. In *International conference on machine learning*, pages 2952–2960. PMLR, 2016.

[19] Fabian Pedregosa. Hyperparameter optimization with approximate gradient. In *International conference on machine learning*, pages 737–746. PMLR, 2016.

[20] Riccardo Grazzi, Luca Franceschi, Massimiliano Pontil, and Saverio Salzo. On the iteration complexity of hypergradient computation. In *International Conference on Machine Learning*, pages 3748–3758. PMLR, 2020.

[21] Mathieu Blondel, Quentin Berthet, Marco Cuturi, Roy Frostig, Stephan Hoyer, Felipe Llinares-López, Fabian Pedregosa, and Jean-Philippe Vert. Efficient and modular implicit differentiation. *Advances in neural information processing systems*, 35:5230–5242, 2022.

[22] Justin Domke. Generic methods for optimization-based modeling. In *Artificial Intelligence and Statistics*, pages 318–326. PMLR, 2012.

[23] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pages 2113–2122. PMLR, 2015.

[24] Ryuichiro Hataya and Makoto Yamada. Nyström method for accurate and scalable implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pages 4643–4654. PMLR, 2023.

[25] Zhongkai Hao, Chengyang Ying, Hang Su, Jun Zhu, Jian Song, and Ze Cheng. Bi-level physics-informed neural networks for pde constrained optimization using broyden's hypergradients. In *The Eleventh International Conference on Learning Representations*, 2022.

[26] Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

[27] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2016.

[28] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[29] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32, 2019.

[30] Hae Beom Lee, Hayeon Lee, Jaewoong Shin, Eunho Yang, Timothy Hospedales, and Sung Ju Hwang. Online hyperparameter meta-learning with hypergradient distillation. *arXiv preprint arXiv:2110.02508*, 2021.

[31] Saeed Ghadimi and Mengdi Wang. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

[32] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[33] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019.

[34] Khurram Javed and Martha White. Meta-learning representations for continual learning. *Advances in neural information processing systems*, 32, 2019.

[35] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.