# Rapport de Stage de Deuxième Année : Autour de l'Optimisation à Deux Niveaux

**Gaspard Choné-Ducasse**
Sous la direction de **Mark Schmidt**
Department of Computer Science
University of British Columbia

## 1 Déroulé du stage

J'ai effectué mon stage de deuxième année de février à juin 2023 au département d'informatique de l'Université de Colombie Britannique à Vancouver sur la côte ouest canadienne. Le stage était supervisé par Mark Schmidt, mais j'ai travaillé principalement avec Chen Fan, doctorant, co-supervisé par Mark et Christos Thrampoulidis.

Mark dirige un laboratoire d'une douzaine d'étudiants en master et de doctorants qui sont spécialisés dans l'optimisation d'algorithmes de machine learning. Les optimisations recherchées sont assez théoriques, même si souvent appuyées par des expériences conclusives. Une implémentation pratique et fonctionnelle n'est pas l'objectif principal.

Avec Chen, nous nous sommes penchés sur le problème du choix des pas de descentes de gradient dans les problèmes d'optimisation à deux niveaux que je décris dans la section qui suit. Nous avons proposé une méthode adaptive construite sur le modèle du pas de Polyak et de *line searches*. Chen a construit une théorie donnant des garanties de convergence et je me suis concentré sur l'expérimentation. J'ai appliqué notre méthode sur deux applications de l'optimisation à deux niveaux, le choix d'hyperparamètres et la distillation de données dont je donne une présentation détaillée plus bas. Ces travaux ont donné lieu à une soumission pour la conférence Neur'IPS.

Ce rapport est construit en deux parties. La première est un résumé simple, en français, des résultats et la seconde est une version abrégée de l'article soumis (afin de rester dans les contraintes de l'exercice).

Ainsi pour des raisons de simplification, nous ne mentionnerons pas dans ce résumé, les difficultés liées au caractère stochastique des algorithmes (ce qui est nécessaire pour appliquer les méthodes à des problèmes modernes). De plus, nous ne traitons pas l'adaptation de nos méthodes à l'algorithme Adam.

## 2 Bilevel Optimization (BO) - Optimization à deux niveaux

L'objectif de l'optimisation à deux niveaux (BO) est de résoudre le problème suivant :

$$\operatorname*{argmin}_{x \in \mathcal{X}} \Phi(x) \;:=\; f(x, y^*(x))$$

où

$$y^*(x) := \operatorname*{argmin}_{y \in \mathcal{Y}} g(x, y).$$

Déterminer $y^*(x)$ est appelé problème interne, déterminer $\operatorname{argmin}_{x \in \mathcal{X}} f(x, y)$ est le problème externe.

Comme pour de nombreux problèmes d'optimisation, la méthode de résolution choisie est basée sur une descente de gradient. Cependant le gradient de $\Phi$ n'est calculable simplement :

$$\nabla\Phi(x) \;=\; \nabla_x f(x, y^*) + \frac{\partial y^*}{\partial x}(x, y^*) \cdot \nabla_y f(x, y^*)$$

où

$$\frac{\partial y^*}{\partial x} = -\nabla^2_{xy} g(x, y^*) \cdot \left( \nabla^2_{yy} g(x, y^*) \right)^{-1}.$$

Les méthodes modernes de dérivation automatique, nous permettent de calculer les dérivés premières et secondes de $f$ et $g$ en tous points, sous des hypothèses raisonnables. Cependant, nous faisons face à deux difficultés : comment obtenir $y^*(x)$ et comment calculer l'inverse de la hessienne.

L'algorithme ci-contre décrit la méthode classique pour résoudre ces problèmes. La méthode suit le schéma d'une descente de gradient d'un problème à un niveau. Cependant, pour calculer l'hypergradient, c'est à dire une approximation de $\nabla\Phi(x)$, à chaque étape, une boucle interne permet d'obtenir une approximation de $y^*(x)$ via une descente de gradient. Puis une approximation de la hessienne est obtenue par les séries de Neumann ou la méthode des gradients conjugués. Nous ne rentrerons pas dans plus de détails ici.

**Algorithm 1** Résolution classique d'un problème de BO

**Input:** $T, N, Q, \alpha, \beta$
**Output:** $x$
$x = 0$
$y = 0$
**for** $t = 1, 2, \dots T$ **do**
  **for** $i = 1, 2, \dots N$ **do**
    $y \leftarrow y - \alpha \cdot \nabla_y g(x, y)$
  **end for**
  Compute an approximation of $\left( \nabla^2_{yy} g(x, y) \right)^{-1} \cdot \nabla_y f(x, y)$ with Q steps of CG or the Neumann Series
  Compute $\widehat{\nabla}\Phi(x)$
  $x \leftarrow x - \beta \cdot \widehat{\nabla}\Phi(x)$
**end for**

## 3 Line Search

Cet algorithme introduit de nombreux hyperparamètres, dont les plus importants : les deux pas de descentes ainsi que la longueur de la boucle interne. Les algorithmes les plus novateurs ne résolvent pas ces problèmes et se concentrent sur l'amélioration de la direction de la descente, par exemple via l'utilisation de momentum.

Cependant, ce problème est majeur comme nous l'avons montré dans la figure 3. En effet, un mauvais choix de pas peut conduire à la divergence de la descente ou une convergence très lente. De plus, ces algorithmes sont très coûteux par l'imbrication des deux boucles. Ainsi, il est difficile d'expérimenter et de tester différentes valeurs de pas possibles.

Nous avons chercher à adapter des méthodes connues de choix adaptif de pas de descentes de gradient : le pas de Polyak et la recherche linéaire de pas, la *line search*. Nous allons ici, nous concentrer sur l'explication de cette dernière.

L'idée d'une *line search* est assez simple. Pour une direction de descente donnée (typiquement l'opposé du gradient), différents pas sont testés afin de garantir une condition assurant le rapprochement d'un minimum. La condition, dite d'Armijo, est la suivante :

$$f(x - \alpha \nabla f(x)) \leq f(x) - c\,\alpha \|\nabla f(x)\|^2$$

Cependant, cette méthode n'est pas directement applicable à notre problème en remplaçant $f$ par $\Phi$. En effet, les quantitées auxquelles nous avons accès après la boucle interne sont :

- $\hat{y}(\mathbf{x})$ := approximation de $y^*(\mathbf{x})$ après $N$ étape de DG
- $\hat{\Phi}_{\mathbf{x}}(x) := f(x, \hat{y}(\mathbf{x}))$
- $\widehat{\nabla}\Phi(x) := \nabla_x f(x, \hat{y}) - \nabla^2_{xy} g(x, \hat{y}) \left( \left( \nabla^2_{yy} g(x, \hat{y}) \right)^{-1} \cdot \nabla_y f(x, \hat{y}) \right)_{\text{CG (Q étapes)}}$

Remarquons de plus que $\nabla\hat{\Phi}_x(x) \neq \widehat{\nabla}\Phi(x)$. Cette différence n'est pas une approximation potentiellement négligeable. En effet, certaines fonctions externes (par exemple dans le cas de la distillation de donnée) n'ont une dépendance qu'en $y$, $\hat{\Phi}_x$ est une fonction constante et alors $\nabla\hat{\Phi}_x(x) = 0$.

Nous écrivons la nouvelle condition :

$$\hat{\Phi}_{x - \beta\widehat{\nabla}\Phi(x)}(x - \beta\widehat{\nabla}\Phi(x)) \leq \hat{\Phi}_x(x) - c\,\beta \|\widehat{\nabla}\Phi(x)\|^2 + \delta$$

Notre solution, présentée dans l'algorithme 2, que nous avons trouvée pour appliquer cette condition, consiste à effectuer un étape de descente de gradient sur la variable $y$ pour pouvoir calculer $\hat{\Phi}_{x - \beta\widehat{\nabla}\Phi(x)}(x - \beta\widehat{\nabla}\Phi(x))$ pour chaque $\beta$ testé.

Le paramètre $\delta > 0$ permet de prendre en compte les erreurs dans le calculs de $\widehat{\nabla}\Phi(x)$ dans les preuves théoriques mais en pratique nous le fixons à 0.

Cet ajout complexifie l'algorithme et augmente le coût de chaque étape, mais le gain obtenu par le choix d'un bon pas de descente est plus important. Nous comparons notre méthode aux algorithmes classiques dans la figure 7.

## 4 Disitillation de données

Le but de la distillation de donnée est d'obtenir une représentation d'un très large jeu de données en une petite sélection. Cette représentation doit ensuite permettre de pouvoir entrainer des modèles avec des performances similaires à un entrainement sur le jeu complet.

Les gains de temps et d'énergie obtenus en entrainant des modèles sur de la donnée plusieurs ordres de grandeur plus petite sont immenses et bien évidemment très intéressants à tous points de vue. La recherche dans ce domaine est très dynamique.

Le problème de la distillation de donnée peut s'exprimer selon un problème d'optimisation à deux niveaux.

Soit $\mathcal{L}_S(w)$ la fonction de *loss* évaluée sur le jeu de donnée $S$ avec les poids $w$ d'un modèle. L'objectif s'écrit :

$$D^* = \underset{D}{\operatorname{argmin}} \mathcal{L}_{\tilde{V}}(w^*(D)) \quad \text{t.q.} \quad w^*(D) = \underset{w}{\operatorname{argmin}} \mathcal{L}_D(w),$$

où $\tilde{V}$ est de la même taille que $D$ et tiré dans le jeu (original) $V$. La solution $D^*$ est la donnée distillée.

L'étape interne est l'entraînement d'un modèle sur de la donnée générée aléatoirement pour commencer, mais qui va devenir la donnée distillée que l'on cherche à extraire. L'étape externe consiste à minimiser la *loss* du modèle tout juste entrainé, calculée sur le jeu de données à distiller. Cependant cette minimisation ne s'effectue pas via la modification des poids du modèle mais par la modification de la donnée interne. Ainsi, cette donnée va converger vers une représentation du jeu de données.

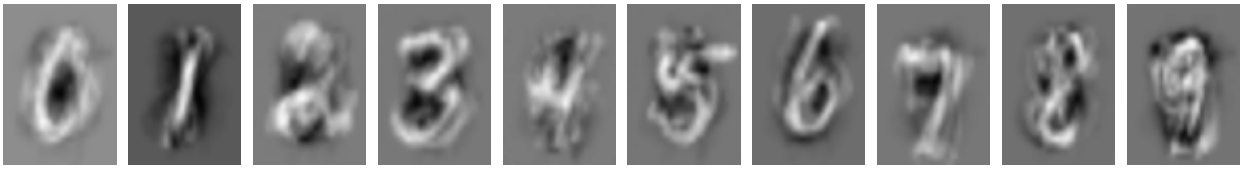
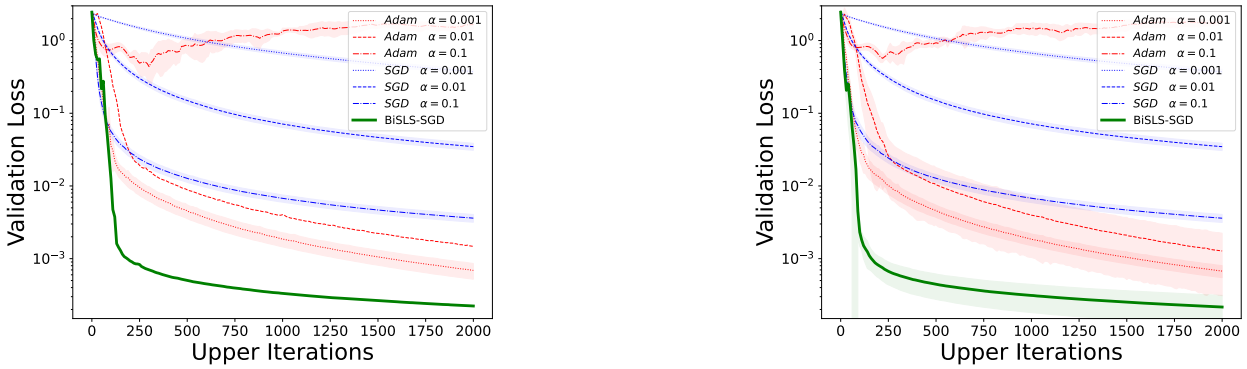


Figure 1: MNIST - Images Distillées

Notre algorithme accélère l'entraînement du modèle et surtout simplifie grandement la tâche pour l'utilisateur en enlevant deux hyperparamètres.




(a) Neumann

(b) Identity

Figure 2: Performance de notre algorithme (BiSLS : Bilevel Stochastic Line Search) contre deux algorithmes classiques Adam et SGD, pour deux méthodes de calcul de la Héssienne.

# BiSLS/SPS: Auto-tune Step Sizes for Stable Bi-level Optimization

**Chen Fan**
Department of Computer Science
University of British Columbia

**Gaspard Choné-Ducasse**
Department of Mathematics and Applications
Ecole Normale Supérieure

**Mark Schmidt**
Department of Computer Science
University of British Columbia
Canada CIFAR AI Chair (Amii)

**Christos Thrampoulidis**
Department of Electrical and Computer Engineering
University of British Columbia

## Abstract

The popularity of bi-level optimization (BO) in deep learning has spurred a growing interest in studying gradient-based BO algorithms. However, existing algorithms involve two coupled learning rates that can be affected by approximation errors when computing hypergradients, making careful fine-tuning necessary to ensure fast convergence. To alleviate this issue, we investigate the use of recently proposed adaptive step-size methods, namely stochastic line search (SLS) and stochastic Polyak step size (SPS), for computing both the upper and lower-level learning rates. First, we revisit the use of SLS and SPS in single-level optimization without the additional interpolation condition that is typically assumed in prior works. For such settings, we investigate new variants of SLS and SPS that improve upon existing suggestions in the literature and are simpler to implement. Importantly, these two variants can be seen as special instances of general family of methods with an envelope-type step-size. This unified envelope strategy allows for the extension of the algorithms and their convergence guarantees to BO settings. Finally, our extensive experiments demonstrate that the new algorithms, which are available in both SGD and Adam versions, can find large learning rates with minimal tuning and converge faster than corresponding vanilla SGD or Adam BO algorithms that require fine-tuning.

## 1 Introduction

Bi-level optimization has found its applications in various fields of machine learning, such as hyperparameter optimization [13, 15, 28, 37], adversarial training [48], data distillation [1, 50], neural architecture search [26, 36], neural-network pruning [49], and meta-learning [12, 34, 10]. Specifically, it is used widely for problems that exhibit a hierarchical structure of the following form:

$$\min_{x\in X} F(x) = \mathbb{E}_{\phi}[f(x, y^*(x); \phi)] \qquad \text{s.t.} \qquad y^*(x) = \operatorname*{argmin}_{y\in Y} \mathbb{E}_{\psi}[g(x, y; \psi)]. \tag{1}$$

Here, the solution to the lower-level objective $g$ becomes the input to the upper-level objective $f$, and in (1) the upper-level variable $x$ is fixed when optimizing the lower-level variable $y$. To solve such bi-level problems using gradient-based methods requires computing the hypergradient of $F$, which based on the chain rule is given as [14]:

$$\nabla F(x) = \nabla_x f(x, y^*(x)) + \nabla^2_{xy} g(x, y^*(x))[\nabla^2_{yy} g(x, y^*(x))]^{-1} \nabla_y f(x, y^*(x)). \tag{2}$$

In practice, the closed-form solution $y^*(x)$ is difficult to obtain, and one strategy is to run a few steps of (stochastic) gradient descent on $g$ w.r.t. $y$ to get an approximation $\bar{y}$, and use $\bar{y}$ in places of $y^*(x)$. We denote the stochastic hypergradient based on $\bar{y}$ as $h_f(x, \bar{y})$ and the stochastic gradient of $g$ w.r.t. $y$ as $h_g$. This leads to a general gradient-based

framework for solving bi-level optimization [14, 17, 3]. At each iteration $k$, run T (can be one or more) steps of SGD on $y$, i.e. $y^{k,t+1} = y^{k,t} - \beta h_g^{k,t}$, then run one step on $x$ using the approximated hypergradient:

$$x^{k+1} = x^k - \alpha h_f(x^k, y^{k+1}), \quad \text{where} \quad y^{k+1} = y^{k,T}. \tag{3}$$

Based on this framework, a series of stochastic algorithms have been developed to achieve the optimal or near-optimal rate of their deterministic counterparts [6, 7]. These algorithms can be broadly divided into single-loop ($T = 1$) or double-loop ($T > 1$) categories [21].

Unlike minimizing the single-level finite-sum (convex) problem

$$F(x) := \min_{x \in C} \frac{1}{N} \sum_{i=1}^{N} f_i(x), \tag{4}$$

where only one learning rate is involved when using SGD, bi-level optimization involves tuning both the lower and upper-level learning rates ($\beta$ and $\alpha$ respectively). This poses a significant challenge due to the potential correlation between these learning rates [17]. Thus, as observed in Figure 3, algorithm divergence can occur when either $\alpha$ or $\beta$ is large. While there is considerable literature on achieving faster rates in bi-level optimization [22, 4, 6, 7], only a few studies have focused on stabilizing its training and automating the tuning of $\alpha$ and $\beta$. This work addresses the question: **Is it possible to utilize large $\alpha$ and $\beta$ without manual tuning?**

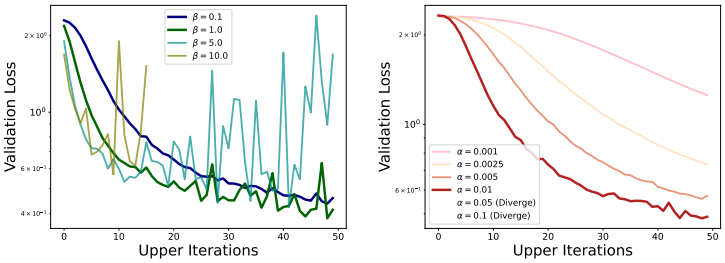



Figure 3: Results based on hyper-representation learning task (see Sec 4 for details). Validation loss against upper-level iterations for different values of $\beta$ (left, $\alpha = 0.005$) and $\alpha$ (right, $\beta = 0.01$). Unless carefully tuned, vanilla SGD-based methods for BO are very unstable.

In doing so, we explore the use of stochastic adaptive-step size methods, namely stochastic Polyak step size (SPS) and stochastic line search (SLS), which utilize gradient information to adjust the learning rate at each iteration [41, 27]. These methods have been demonstrated to perform well in interpolation settings with strong convergence guarantees [41, 27]. However, applying them to bi-level optimization (BO) introduces significant challenges, as follows. ① BO requires tuning two correlated learning rates (for lower and upper-level). ② The bias in the stochastic approximation of the hypergradient complicates the practical performance and convergence analysis of SLS and SPS. ③ Other algorithmic challenges arise for both algorithms: For SLS, verifying the stochastic Armijo condition at the upper-level involves evaluating the objective at a new $(x, y^*(x))$ pair, while $y^*(x)$ is only approximately known; For SPS, most existing variants guarantee good performance only in interpolating settings, which are typically not satisfied for the upper-level objective in BO [20]. Before presenting our solutions to the challenges above in Sec 2, we first review the most closely related literature.

**Algorithm 2** BiSLS-Adam/SGD

**Input:** $x^0$, $y^0$, $K$, $T$, $\delta$, $\alpha_{b,0}$, $\beta_{b,0}$, $w$, $\eta$
**Output:** $x$

**for** $k = 0, 1, \ldots, K-1$ **do**
  $y^{k,0} = y^k$
  **for** $t = 0, 1, \ldots, T-1$ **do**
    $\beta_{b,k}^t \leftarrow \text{reset}(\beta, \beta_{b,0}, \eta, \text{opt})$ ▷ see Algorithm 3
    $\beta \leftarrow$ linesearch based on (8) starting from $\beta_{b,k}^t$
    $y^{k,t+1} = y^{k,t} - \beta\, h_g^{k,t}$,
  **end for**
  $y^{k+1} = y^{k,T-1}; \hat{x}^k = x^k$; $\hat{y}^{k+1} = y^{k+1}$
  $\alpha \leftarrow \text{reset}(\alpha, \alpha_{b,0}, \eta, \text{opt})$
  **while** (14) based on $(\hat{x}^k, \hat{y}^{k+1}, \alpha, \delta)$ does not hold. **do**
    $\alpha = \alpha * w$
    $\hat{x}^k = x^k - \alpha\, h_f(x^k, y^{k+1})$ or
    $\hat{x}^k = x^k - \alpha A_k^{-1} h_f(x^k, y^{k+1})$
    $\hat{y}^{k+1} = y^{k+1} - \beta\, h_g(\hat{x}^k, y^{k+1})$
  **end while**
  $x^{k+1} = x^k - \alpha\, h_f^k$
**end for**

### 1.1 Related Work

**Gradient-Based Bi-level Optimization** Penalty or gradient-based approaches have been used for solving bi-level optimization problems [9, 42, 19]. Here we focus our discussions on stochastic gradient-based methods as they are closely related to this work. For double-loop algorithms, an early work (BSA) by Ghadimi and Wang [14] has derived the sample complexity of $\phi$ in achieving an $\epsilon$-stationary point to be $O(\epsilon^{-2})$, but require the number of lower-level steps to satisfy $T \sim O(\epsilon^{-1})$. Using a warm start strategy (stocBiO), Ji et al. [20] removed this requirement on $T$. However, to achieve the same sample complexity, the batch size of stocBiO grows as $O(\epsilon^{-1})$. Chen et al. [3] removed both requirements on $T$ and batch size by using the smoothness properties of $y^*(x)$ and setting the step sizes $\alpha$ and $\beta$ at the same scale. For single-loop algorithms, a pioneering work by Hong et al. [17] gave a sample complexity of $O(\epsilon^{-2.5})$,

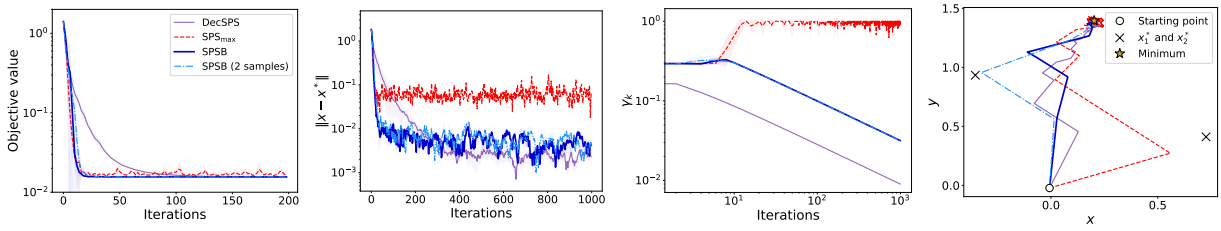



Figure 4: Experiments on quadratic functions adapted from [27]. The objective is the sum of two-dimensional functions $f_i = \frac{1}{2}(x - x_i^*)^T H_i (x - x_i^*)$, where $H_i$ is positive definite and $i = 1, 2$ (see Appendix A for more details). From left to right, we show: the objective value, distance to optimum, step size, and iterate trajectories.

provided $\alpha$ and $\beta$ are on two different scales (TTSA). By making corrections to the $y$ variable update (STABLE), Chen et al. [4] improved the rate to $O(\epsilon^{-2})$. However, extra matrix projections required by STABLE can incur high computation cost [4, 3]. By incorporating momentum into the updates of $x$ and $y$ (SUSTAIN), Khanduri et al. [22] further improved the rate to $O(\epsilon^{-1.5})$ [5]. Besides these single or double-loop algorithms, a series of works have drawn ideas from variance reduction to achieve faster convergence rates for BO. For example, Yang et al. [46] designed the VRBO algorithm based on SPIDER [11]. Dagréou et al. [6, 7] designed the SABA and SRBA algorithms based on SAGA and SARAH respectively, and demonstrate that they can achieve the optimal rate of $O(\epsilon^{-1})$ [8, 32].

Huang et al. [18] proposes to use Adam-type step sizes in BO. However, it introduces three sequences of learning rates $(\alpha_k, \beta_k, \eta_k)$ that require tuning, which limits its practical usage. To our knowledge, none of these works have explicitly addressed the fundamental problem of how to select $\alpha$ and $\beta$ in bi-level optimization. In this work, we focus on the alternating SGD framework (T can be 1 or larger), and design efficient algorithms that find large $\alpha$ and $\beta$ without tuning, while ensuring the stability of training.

**Adaptive Step Size** Adaptive step-size such as Adam has found great success in modern machine learning, and different variants have been proposed [23, 35, 44, 29, 30]. Here, we limit our discussions on two adaptive step sizes that are most relevant to this work. The Armijo line search is a classic way for finding step sizes for gradient descent [45]. Vaswani et al. [41] extends it to the stochastic setting (SLS) and demonstrates that the algorithm works well with minimal tuning required under interpolation, where model fits the data perfectly. Hence, the method is adaptive to local smoothness of the objective, which is typically difficult to predict a priori. However, the theoretical guarantee of SLS in the non-interpolating regime is lacking. In fact, the results in Figure 5 suggest that SLS can perform poorly for convex losses when interpolation is not satisfied. Besides SLS, another adaptive method derived from Polyak step size is proposed by Loizou et al. [27] with the name stochastic Polyak step size (SPS). Loizou et al. [27] further places an upper bound on the step size resulting in the $\text{SPS}_{\max}$ variant. Similar to SLS, the algorithm performs well when the model is over-parametrized. Without interpolation, the algorithm converges to a neighborhood of the solution whose size depends on this upper bound. In a later work, Orvieto et al. [33] make the SPS converge to the exact solution by ensuring the step size and its upper bound are both non-increasing (DecSPS ). However, enforcing monotonicity may result in the step size being smaller than decaying-step SGD and losing the adaptive features of SPS (see Figure 4, 5). In this work, we propose new versions of SLS and SPS that do not require monotonicity and extend them into the alternating SGD bi-level optimization framework (3).

## 2 Summary of Contributions

We discuss our main contributions in this section, which is organized as follows. First, we discuss our variants of SPS and SLS, and unify them under the umbrella of "envelope-type step-size". Then, we extend the envelope-type step size to the bi-level setting. Finally, we discuss our bi-level line-search algorithms based on Adam and SGD.

**Converging SPSB and SLSB by Envelope Approach** We first propose simple variants of SLS and SPS that converge in the non-interpolating setting while not requiring the step size to be monotonic. To this end, we introduce a new stochastic Polyak step size (SPSB). For comparison, we also recall the step-sizees of $\text{SPS}_{\max}$ and DecSPS . For all methods, the iterate updates are given as $x_{k+1} = x_k - \gamma_k \nabla f_{i_k}(x^k)$ where $i_k$ is sampled uniformly from $[n] = \{1, \ldots, n\}$

at each iteration $k$. The step-sizes $\gamma_k$ are then defined as follows:

$$\text{SPS}_{\max}\ [27]: \quad \gamma_k = \min\{\frac{f_{i_k}(x^k) - f_{i_k}^*}{c\|\nabla f_{i_k}(x^k)\|^2}, \gamma_{b,0}\} \tag{5}$$

$$\text{DecSPS}\ [33]: \quad \gamma_0 = \bar{\gamma} \quad \gamma_k = \frac{1}{c_k}\min\{\frac{f_{i_k}(x^k) - l_{i_k}^*}{\|\nabla f_{i_k}(x^k)\|^2}, c_{k-1}\gamma_{k-1}\} \quad \forall k \geq 1 \tag{6}$$

$$\textbf{SPSB (ours):} \quad \gamma_k = \min\{\frac{f_{i_k}(x^k) - l_{i_k}^*}{c_k\|\nabla f_{i_k}(x^k)\|^2}, \gamma_{b,k}\}, \tag{7}$$

where $f_i^* = \inf_x f_i(x)$, $\bar{\gamma} = \frac{1}{c_0}\min\{\frac{f_{i_0}(x^0) - l_{i_0}^*}{\|\nabla f_{i_0}(x^0)\|^2}, c_0\gamma_{b,0}\}$ , $c_k$ is non-decreasing, $\gamma_{b,k}$ is non-increasing, and $l_i^* \leq f_i^*$ is any lower bound.

Unlike $\text{SPS}_{\max}$ in which $\gamma_{b,0}$ is a constant, our upper bound $\gamma_{b,k}$ is non-increasing. Also, unlike DecSPS in which both the step size and the upper bound are non-increasing (this is because $\gamma_k \leq \frac{c_{k-1}}{c_k}\gamma_{k-1}$ and $\min\{\frac{1}{2cL_{\max}}, \frac{c_0\gamma_{b,0}}{c_k}\} \leq \gamma_k \leq \frac{c_0\gamma_{b,0}}{c_k}$ [33, Lemma 1]), we simplify the recursive structure and do not require the step-size to be monotonic. As we empirically observe in Figure 5, the step size of DecSPS is similar to that of decaying SGD and in fact can be much smaller. Interestingly, the resulting performance of DecSPS is worse than $\text{SPS}_{\max}$ despite $\text{SPS}_{\max}$ eventually becoming unstable once iterates get closer to the neighborhood of a solution and the step-size naturally behaves erratically. This is not unexpected due to small gradient norms (note division by gradient-norm in (5)) and dissimilarity between samples in the non-interpolating scenario. Moreover, note that the adaptivity of SPS in the early stage seems to be lost in DecSPS due to monotonicity of the latter. On the other hand, SPSB not only takes advantage of the large SPS steps that leads to fast convergence, but also stays regularized due to the non-increasing upper bound $\gamma_{b,k}$ in (7). These observations are further supported by the experiments on quadratic functions given in Figure 4, where we observe the fast convergence of SPSB and the instability of $\text{SPS}_{\max}$ . Motivated by the good practical performance of SPSB , we take a similar approach for SLS. The SLS proposed and analyzed by Vaswani et al. [41] starts with $\gamma_{b,0}$ and in each iteration $k$ finds the largest $\gamma_k \leq \gamma_{b,0}$ that satisfies:

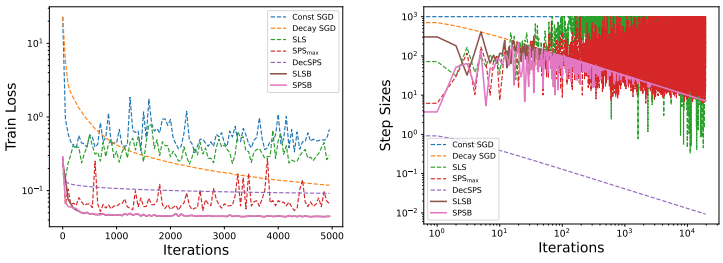



Figure 5: Binary linear classification on w8a dataset using logistic loss [2]. We choose $\gamma_{b,0} = 1000$ for all algorithms; $c = 1$ and $\bar{c} = 1$ for $\text{SPS}_{\max}$ and SLS respectively; $c_k = \sqrt{k+1}$ for DecSPS ; $c_k = 1$ and $\gamma_{b,k} = \frac{\gamma_{b,0}}{\sqrt{k+1}}$ for SPSB ; $\bar{c} = 0.1$ and $\gamma_{b,k} = \frac{\gamma_{b,0}}{\sqrt{k+1}}$ for SLSB ; $\gamma_{b,k} = \frac{\gamma_{b,0}}{\sqrt{k+1}}$ for decaying-step SGD.

$$f_{i_k}(x_k - \gamma_k\nabla f_{i_k}(x_k)) \leq f_{i_k}(x_k) - \bar{c}\cdot\gamma_k\|\nabla f_{i_k}(x_k)\|^2, \quad 0 < \bar{c} < 1. \tag{8}$$

To ensure its convergence without interpolation, we replace $\gamma_{b,0}$ with appropriate non-increasing sequence $\gamma_{b,k}$. We name this variant of SLS as SLSB . Interestingly, the empirical performance and step size of SLSB are similar to those of SPSB (see Figure 5). This can be explained by observing that the step sizes of SPSB and SLSB share similar envelope structures, as follows:

$$\text{SPSB}: \quad \min\{\frac{1}{2cL_{\max}}, \gamma_{b,k}\} \leq \gamma_k = \min\{\frac{f_{i_k}(x^k) - l_{i_k}^*}{c\|\nabla f_{i_k}(x^k)\|^2}, \gamma_{b,k}\}, \quad 0 < c,$$

$$\text{SLSB}: \quad \min\{\frac{2(1-\bar{c})}{L_{\max}}, \gamma_{b,k}\} \leq \gamma_k \leq \min\{\frac{f_{i_k}(x^k) - l_{i_k}^*}{\bar{c}\|\nabla f_{i_k}(x^k)\|^2}, \gamma_{b,k}\}, \quad 0 < \bar{c} < 1.$$

Therefore, we unify their analysis based on the following generic *envelope-type step size*:

$$\gamma_k = \min\{\max\{\gamma_{l,k}, \tilde{\gamma}_k\}, \gamma_{b,k}\}, \quad \gamma_{l,k} = \min\{w, \gamma_{b,k}\}, \tag{9}$$

where $\omega > 0$, $\gamma_{b,k}$ is non-increasing, and $\tilde{\gamma}_k$ satisfies $\gamma_{l,k} := \min\{\omega, \gamma_{b,k}\} \leq \tilde{\gamma}_k \leq \gamma_{b,k}$. We show that this envelope-type step size converges at a rate $O(\frac{1}{\sqrt{K}})$ and $O(\frac{1}{K})$ for convex and strongly-convex losses respectively.

**Envelope Step Size for Bi-level Optimization (BiSPS)** We extend the analysis of envelope-type step size to the bi-level setting. The step sizes for upper and lower-level objectives of our general envelope-type method are:

$$\text{Upper: } \alpha_k = \min\{\max\{\alpha_{l,k}, \tilde{\alpha}_k\}, \alpha_{b,k}\} \quad \text{hence} \quad \alpha_{l,k} \leq \tilde{\alpha}_k \leq \alpha_{b,k} \tag{10}$$

$$\text{Lower: } \beta_{k,t} = \min\{\frac{g(x^k, y^{k,t}; \psi) - g(x^k, y^*_{x^k,\psi}; \psi)}{p\|\nabla_y g(x^k, y^{k,t}; \psi)\|^2}, \beta_{b,k}\} \quad \forall t, \tag{11}$$

where $y^*_{x^k,\psi}$ is the minimizer of the function $g(x^k, \cdot; \psi)$, and $\alpha_{l,k}$, $\alpha_{b,k}$, and $\beta_{b,k}$ are three non-increasing sequences. Note that $\beta_{b,k}$ is fixed over the lower-level iterations for a given $k$, therefore, this is equivalent to running $T$ steps of $\text{SPS}_{\max}$ to minimize the function $g$ at each upper iteration $k$. However, the decrease in the upper bound $\beta_{b,k}$ with $k$ is crucial to guarantee the overall convergence of the algorithm (see Theorem 3). Starting from the general step-size rules in (10), (11), our bi-level extension of SPS, which we call BiSPS, follow by setting $\alpha_k$ in the form of SPS computed using stochastic hypergradient $h^k_f$. That is,

$$\bar{\alpha}_k = \frac{f(x^k, y^{k+1}; \phi) - l^*_{f(\cdot, y^{k+1}; \phi)}}{p\|h^k_f\|^2}, \quad \alpha_{l,k} = \frac{\alpha_{l,0}}{\sqrt{k+1}}, \quad \alpha_{b,k} = \frac{\alpha_{b,0}}{\sqrt{k+1}}, \tag{12}$$

where $\alpha_{l,0} \leq \alpha_{b,0}$ and $l^*_{f(\cdot, y^{k+1}; \phi)}$ is a lower bound for $\inf_x f(x, y^{k+1}; \phi)$. For computing $h^k_f$, we can take a similar approach as previous works [14, 17, 3] that use Neumann series setting

$$h^k_f = \nabla_x f(x^k, y^{k+1}; \phi) - \nabla_{xy} g(x^k, y^{k+1}; \psi_0)\Big[\frac{N}{L_g}\prod_{j=1}^{\bar{N}}(I - \nabla^2_{yy} g(x^k, y^{k+1}; \psi_j))\Big]\nabla_y f(x^k, y^{k+1}; \phi), \tag{13}$$

where $\bar{N}$ is sampled uniformly from $[N]$ and $N$ is the total number of samples.

For BiSPS, we use the same sample for $f(x^k, y^{k+1}; \phi)$ and $\nabla f_x(x^k, y^{k+1}; \phi)$ when evaluating $\bar{\alpha}_k$ in (12). Interestingly, we also empirically observe that using independent samples for computing $\bar{\alpha}_k$ and $h^k_f$ resulting in similar performance as using the same sample. The optimal rate of SGD for non-convex bi-level optimization is $O(\frac{1}{\sqrt{K}})$ without a growing batch size [3]. We show that BiSPS can obtain the same rate (see Theorem 3) by taking the envelope-type step-size of the form (10) and (11). We implement BiSPS according to (12) and observe that it has better performances over decaying-step SGD with less variations across different values of $\alpha_{b,0}$ (see Figure 6 and note that decaying-step SGD is of the form $\frac{\alpha_{b,0}}{\sqrt{k+1}}$).

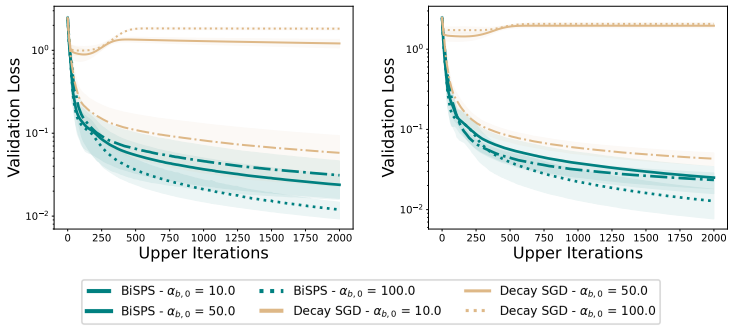
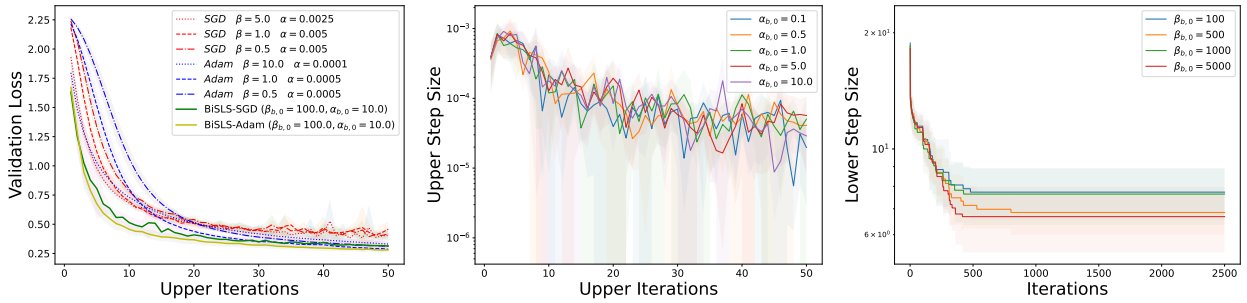



Figure 6: Results on data distillation experiments adapted from Lorraine et al. [28] (see Sec 4 for details). We compare BiSPS and decaying-step SGD for different values of $\alpha_{b,0}$ where Hessian inverse in (2) is computed based on the Identity matrix (left) or Neumann series (right). The lower-level learning rate is fixed at $10^{-4}$.

**Stochastic Line-Search Algorithms for Bi-level Optimization** The challenge of extending SLS to bi-level optimization is rooted in the term $y^*(x)$. In fact, we realize that some of the bi-level objectives are of the form $F(x) = f(y^*(x))$. That is, $f$ does not have an explicit dependence on $x$, e.g. the data hyper-cleaning task [20]. This implies that when SLS takes a potential step on $x$, the approximation of $y^*(x)$ (i.e, $\bar{y}(x)$) also needs to be updated, otherwise there is no change in function values. Moreover, the use of approximation $\bar{y}(x)$ and the stochastic estimation error in hypergradient would not gaurantee a step size can be always found. To this end, we modify the Armijo line-search rule to be:

$$\begin{aligned} &\text{BiSLS-SGD:} \quad f\big(x^k - \alpha_k h^k_f, \hat{y}^{k+1}(x^k - \alpha_k h^k_f)\big) \leq f(x^k, y^{k+1}) - p\alpha_k\|h^k_f\|^2 + \delta, \\ &\text{BiSLS-Adam:} \quad f\big(x^k - \alpha_k A_k^{-1} h^k_f, \hat{y}^{k+1}(x^k - \alpha_k A_k^{-1} h^k_f)\big) \leq f(x^k, y^{k+1}) - p\alpha_k\|h^k_f\|^2_{A_k^{-1}} + \delta, \end{aligned} \tag{14}$$

where $p, \delta > 0$ and $A_k$ is a positive definite matrix such that $A^2_k = G_k$. Similar to the single-level Adam case, the matrix $G_k$ in the bi-level setting is defined as $G_k = (\beta_2 G_{k-1} + (1-\beta_2)\,\text{diag}(h^k_f {h^k_f}^T))/(1-\beta^k_2)$ [23, 40]. Moreover, BiSLS-Adam




(a) Validation Loss (b) Upper-level step sizes (c) Lower-level step sizes

Figure 7: Results on hyper-representation learning task (see Sec 4 for details). (a) Validation loss against upper-level iterations for comparing BiSLS-Adam/SGD to fine-tuned Adam/SGD. (b)(c) Upper (left) and lower-level (right) learning rates found by BiSLS-Adam. For the tuned Adam, the optimal lower and upper-level learning rates are $O(1)$ and $O(10^{-4})$, respectively. BiSLS-Adam outperforms tuned Adam/SGD with a starting point that is 5 orders of magnitude larger than the optimal step size.

takes the following steps for updating the variable $x$: $x^{k+1} = x^k - \alpha_k A_k^{-1} m_k$ where $m^{k+1} = \beta_1 m^k - (1-\beta_1) h_f^k$. The details are given in Algorithms 2 and 3. We denote the search starting point for the upper-level as $\alpha_{b,k}$ at iteration $k$, and denote it as $\beta_{b,k}^t$ at step $t$ within iteration $k$ for the lower-level. We remark the following key benefits of resetting $\alpha_{b,k}$ and $\beta_{b,k}^t$ (by using Algorithm 3) to larger values with reference to $\alpha_k$ and $\beta_k^t$ (respectively) at each step: (1) Avoid always searching from $\alpha_{b,0}$ or $\beta_{b,0}^0$, thus, reducing computation cost, and, (2) preserving an overall non-increasing (not necessarily monotonic) trend for $\alpha_{b,k}$ and $\beta_{b,k}^t$, thus, improving training stability. We found different values of $\eta$ all work well (see Appendix A).

The key algorithmic challenge we are facing is that during the backtracking process, for any candidate $\alpha_k$, we need to compute $\hat{x}^k := x^k - \alpha_k h_f^k$ and approximate $y^*(\hat{x}^k)$ with $\hat{y}^{k+1}$ (see Algorithm 2).

To limit the cost induced by this nested loop, we limit the number of steps to obtain $\hat{y}^{k+1}$ to be 1.

Moreover, $\delta$ in (14) plays the role of a safeguard that ensures a step size can be found.

We set it to be small to avoid finding unrealistically large learning rates while tolerating some error in the hypergradient estimation (see Appendix A for experiments on the sensitivity of $\delta$). In practice, we empirically find that simply setting $\delta = 0$ works well. In Figure 7a, we observe that BiSLS-Adam outperforms fine-tuned Adam or SGD. Surprisingly, its training is stable even when the search starting point $\alpha_{b,0}$ is 5 orders of magnitude larger than a fine-tuned learning rate ($O(10^{-4})$). Importantly, BiSLS-Adam finds large upper and lower-level learning rates in early phase (see Figure 7b, 7c) for different values of $\alpha_{b,0}$ and $\beta_{b,0}$ that span 3 orders of magnitudes. Interestingly, the learning rates naturally decay with training (also see Figure 8c and 8d). In essence, BiSLS is a **truly adaptive (no knowledge of initialization required) and robust (different initialization works) method that finds large $\alpha$ and $\beta$ without tuning**. In the next section, we give the convergence results of the envelope-type step size.

**Algorithm 3** reset

**Input:** $p, q, \eta \geq 1$, opt
**Output:** $p$
1: **if** opt = 1 **then**
2: $p \leftarrow q$
3: **else if** opt = 2 **then**
4: $p \leftarrow p$
5: **else if** opt = 3 **then**
6: $p \leftarrow \eta \cdot p$
7: **end if**

## 3 Convergence Results

### 3.1 Envelope-type step size for single-level optimization

We first state the assumptions, which are standard in the literature, that will be used for analyzing single-level problems. Assumption 1 is on the Lipschitz continuity of $f$ and $f_i$ in Problem 4.

**Assumption 1.** *The individual function $f_i$ is convex and $L_i$-smooth such that $\|\nabla f_i(x) - \nabla f_i(x')\| \leq L_i \|x - x'\|, \forall i, \forall x \in \text{dom } f$ and the overall function $f$ is $L$-smooth. We denote $L_{\max} \triangleq \max_i L_i$. Furthermore, we assume there exists $l_i^*$ such that $l_i^* \leq f_i^* := \inf_x f_i(x), \forall i$, and $f$ is lower bounded by $f^*$ obtained by some $x^*$ such that $f^* = f(x^*)$.*

The following bounded gradient assumption is also used in the analysis of convex problems [38, 31].

**Assumption 2.** *There exists $G > 0$ such that $\|\nabla f_i(x)\|^2 \leq G, \forall i$.*

We first state the theorem for the envelop-type step size defined in (9) for convex functions.

**Theorem 1.** *Suppose Assumption 1, 2 hold, each $f_i$ is convex, $C = \mathrm{dom}\, f$, $\gamma_k$ is independent of the sample $\nabla f_k(x^k)$, and choose $\gamma_{b,k} = \frac{\gamma_{b,0}}{\sqrt{k+1}}$. Then, the envelope-type step size in* (9) *achieves the following rate,*

$$\mathbb{E}[f(\bar{x}^K) - f(x^*)] \leq \frac{\|x^0 - x^*\|^2}{2\gamma_{l,K-1}K} + \frac{\gamma_{b,0}^2 G^2 \log(K)}{2\gamma_{l,K-1}K},$$

*where $\gamma_{l,K-1} = \min\{\omega, \frac{\gamma_{b,0}}{\sqrt{K}}\}$ and $\bar{x}^K = \frac{1}{K}\sum_{k=0}^{K} x^k$.*

We were not able to give a convergence result that uses the same sample for computing the step size and the gradient. However, we empirically observe that the performance is very similar when using either one or two independent samples per iteration (see Figure 4 and Appendix A). When two independent samples $i_k$ and $j_k$ are used per iteration, the first computes the gradient sample $\nabla f_{i_k}(x^k)$, and the other computes the step-size $\gamma_k$. For example, for SPSB this gives $\gamma_k = \min\{\frac{f_{j_k}(x^k) - l^*_{j_k}}{c_k \|\nabla f_{j_k}(x^k)\|^2}, \gamma_{b,k}\}$. This type of assumption has been used in several other works for analyzing adaptive step sizes [25, 41, 27]. Under this assumption, we specialize the results of Theorem 1 to SPSB and SLSB , where $\gamma_{l,K-1} = \min\{\frac{1}{2cL_{\max}}, \frac{\gamma_{b,0}}{\sqrt{K}}\}$ and $\gamma_{l,K-1} = \min\{\frac{2(1-\bar{c})}{L_{\max}}, \frac{\gamma_{b,0}}{\sqrt{K}}\}$ respectively. Concretely, for $K \geq \gamma_{b,0}^2 L_{\max}^2$, SLSB and SPSB with $\gamma_{b,k} = \frac{\gamma_{b,0}}{\sqrt{k+1}}$ and $c = \bar{c} = \frac{1}{2}$ achieve the following rate:

$\mathbb{E}[f(\bar{x}^K) - f(x^*)] \leq \frac{\|x^0 - x^*\|^2}{2\gamma_{b,0}\sqrt{K}} + \frac{\gamma_{b,0} G^2 \log(K)}{2\sqrt{K}}$.

Next, we state the result for the envelop-type step size when $f$ is $\mu$-strongly convex.

**Theorem 2.** *Suppose a $\mu$-strongly convex function $f$ satisfying Assumptions 1 and 2, assume $C$ is a closed and convex set, and $\gamma_k$ is independent of the sample $\nabla f_k(x^k)$. Then an envelope-type step size as in* (9) *with $\gamma_{b,k} = \frac{\gamma_{b,0}}{k+1}$, $\gamma_{b,0} \geq \frac{1}{\mu}$, and $\omega\mu < 1$ achieves the following rate*

$$\mathbb{E}[f(\bar{x}_K) - f(x^*)] \leq \frac{\mu k_0}{2(K-k_0)}\left(e^{-k_0\mu\omega}\|x_0 - x^*\|^2 + \gamma_{b,0}^2 G^2\right) + \frac{\gamma_{b,0} G^2 \log K}{2(K-k_0)},$$

*where $\bar{x}_K = \frac{1}{K-k_0}\sum_{k=k_0}^{K-1} x^k$ and $k_0 = \max\{1, \lceil \gamma_{b,0}/\omega \rceil - 1\}$.*

We can again apply the result of Theorem 2 to SPSB and SLSB with $\gamma_{b,k} = \frac{\gamma_{b,0}}{k+1}$, $\gamma_{b,0} \geq \frac{1}{\mu}$, $\omega = 1/L_{\max}$, and $c = \bar{c} = \frac{1}{2}$ to get an explicit rate: $\mathbb{E}[f(\bar{x}_K) - f(x^*)] \leq \frac{\mu k_0}{2(K-k_0)}\left(e^{\frac{-k_0\mu}{L_{\max}}}\|x_0 - x^*\|^2 + \gamma_{b,0}^2 G^2\right) + \frac{\gamma_{b,0} G^2 \log K}{2(K-k_0)}$, where $k_0 = \max\{1, \lceil \gamma_{b,0} L_{\max} \rceil - 1\}$.

**Remark 1.** *Under the envelop-type step size framework and the assumption of two independent samples, SLSB and SPSB share the same convergence rates of $O(\frac{1}{\sqrt{K}})$ and $O(\frac{1}{K})$ as SGD with decaying step-size for convex and strongly-convex losses respectively. This is not surprising because of the structure of the envelope step-size in* (9)*. Indeed, the proof is similar to the standard proof of analogous rate for SGD with decaying step-size. Nonetheless, we include it here for completeness.*

### 3.2 Envelope-type step size for bi-level optimization

We start with recalling standard assumptions in BO [20, 17, 14, 3]. We denote $z = [x; y]$ and recall the bi-level problem in (1). The first assumption is on the lower-level objective $g$.

**Assumption 3.** *The function $g(x, y)$ is $\mu_g$ strongly convex in $y$ for any given $x$. Moreover, $\nabla g$ is Lipschitz continuous: $\|\nabla g(x_1, y_1) - \nabla g(x_2, y_2)\| \leq L_g\|z_1 - z_2\|$ (also assume that this holds true for each sampled function $g(x, y; \psi)$), and $\nabla^2 g$ is Lipschitz continuous: $\|\nabla^2 g(x_1, y_1) - \nabla^2 g(x_2, y_2)\| \leq L_G\|z_1 - z_2\|$. We further assume that $\|\nabla^2_{xy} g(x, y)\| \leq C_g$, and the condition number is defined as $\kappa = \frac{L_g}{\mu_g}$.*

Next, we state the assumptions on the upper objective $f$.

**Assumption 4.** *The function $f$ and its gradients are Lipschitz continuous. That is: $\|f(x_1, y_1) - f(x_2, y_2)\| \leq L_1\|z_1 - z_2\|$ and $\|\nabla f(x_1, y_1) - \nabla f(x_2, y_2)\| \leq L_{f,1}\|z_1 - z_2\|$. We also assume that $\|\nabla_y f(x, y)\| \leq C_f$.*

Furthermore, we make the following standard assumptions on the estimates of $\nabla f$, $\nabla g$, and $\nabla^2 g$.

**Assumption 5.** *The stochastic gradients are unbiased:* $\mathbb{E}_\phi[\nabla f(x,y;\phi)] = \nabla f(x,y)$, $\mathbb{E}_\psi[\nabla g(x,y;\psi)] = \nabla g(x,y)$, *and* $\mathbb{E}_\psi[\nabla^2 g(x,y;\psi)] = \nabla^2 g(x,y)$. *The variances of* $\nabla f(x,y;\phi)$ *and* $\nabla^2 g(x,y;\psi])$ *are bounded:* $\mathbb{E}_\phi[\|\nabla f(x,y;\phi) - \nabla f(x,y)\|^2] \leq \sigma_f^2$ *and* $\mathbb{E}_\psi[\|\nabla^2 g(x,y;\psi) - \nabla^2 g(x,y)\|^2] \leq \sigma_G^2$.

Finally, we introduce the bounded optimal function value assumption in (15), which is used specifically for analyzing step size of the form (11) in the bi-level setting:

$$\mathbb{E}_\psi[g(x,y^*(x);\psi) - g(x,y^*_{x,\psi};\psi)] \leq \sigma_g^2, \forall x, \tag{15}$$

$$\mathbb{E}[\|\nabla_y g(x,y) - \nabla_y g(x,y;\phi)\|^2] \leq \sigma_g^2, \forall x, y, \tag{16}$$

where $y^*(x) = \min_y g(x,y)$ and $y^*_{x,\psi} = \min_y g(x,y;\psi)$ for a given $x$ (recall that at any iteration $k$, the lower-level steps in BiSPS are $\text{SPS}_{\max}$ with an upper bound $\beta_{b,k}$; furthermore, $\beta_{b,k}$ is non-increasing w.r.t. upper iteration $k$). The one-variable analogous assumption of (15) has been used in the analysis of $\text{SPS}_{\max}$ [27]. Here, we extend it to a two-variable function. Unlike the bounded variance assumption (16), which needs to hold true for all $x$ and $y$, we require (15) to hold at $y^*(x)$ for any given $x$. As mentioned previously, the closed form solution $y^*(x)$ is difficult to obtain. Thus, we define the following expression by replacing $y^*(x)$ with $y$ in (2):

$$\bar{\nabla} f(x,y) = \nabla_x f(x,y) + \nabla^2_{xy} g(x,y)[\nabla^2_{yy} g(x,y)]^{-1} \nabla_y f(x,y). \tag{17}$$

A stochastic Neumann series in (13) approximates (17) with $x$ and $y$ being $x^k$ and $y^{k+1}$ (respectively), also recall that $y^{k+1}$ is an approximation of $y^*(x^k)$ by running $T$ lower-level SGD steps to minimize $g$ w.r.t. $y$ for a fixed $x^k$. Based on Assumptions 3, 4, and 5, we have the following results to be used in the analysis [14]: $\|\nabla F(x_1) - \nabla F(x_2)\| \leq L_F\|x_1 - x_2\|$, $\|y^*(x_1) - y^*(x_2)\| \leq L_y\|x_1 - x_2\|$, and $\|\bar{\nabla} f(x,y^*(x)) - \bar{\nabla} f(x,y)\| \leq L_f\|y^*(x) - y\|$. Furthermore, the bias in the stochastic hypergradient in (13) (denoted as $B$) decays exponentially with $N$ and its variance is bounded, i.e. $\mathbb{E}[\|h_f^k - \mathbb{E}[h_f^k]\|^2] \leq \tilde{\sigma}_f^2$. [17].

Now, we state our main theorem based on step size of the form (10) and (11).

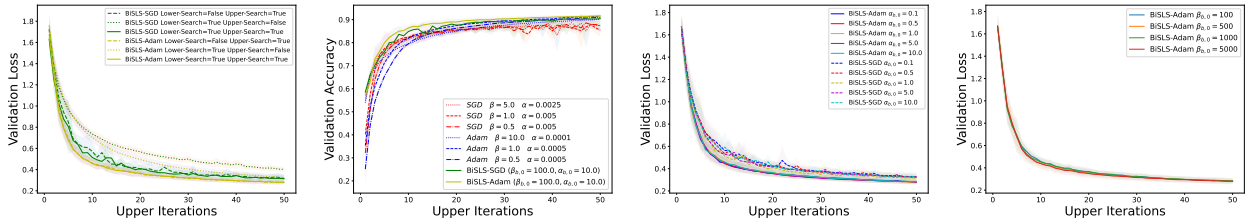
**Theorem 3.** *Suppose* $f$ *and* $g$ *satisfy Assumptions 3, 4, and 5, learning-rate upper bounds* $\alpha_{b,k} = \frac{\alpha_{b,0}}{\sqrt{k+1}}$ *and* $\alpha_{l,k} = \frac{\alpha_{l,0}}{\sqrt{k+1}}$ *with* $\alpha_{b,0}$ *and* $\alpha_{l,0}$ *satisfying* $\frac{1}{L_F + 4L_y^2} \geq \frac{\alpha_{b,0}^2}{\alpha_{l,0}}$ *and* $\alpha_{l,0} \leq \alpha_{b,0}$. *Further assume that* $\alpha_k$ *is independent of the stochastic hypergradient* $h_f^k$, *and each sampled function* $g(x,y;\psi)$ *is convex. Then under the Assumption* (15) *with* $p \geq \frac{1}{2}$, $C_k = \min\{\frac{1}{2pL_g}, \beta_{b,k}\}$, $T \geq \frac{\log(\alpha_{b,0}L_f^2 + 2)}{-\log(1-\mu_g C_{K-1})}$, *and* $\beta_{b,k} = \frac{\beta_{b,0}}{k+1}$, *BiSPS achieves the rate:*

$$\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}[\|\nabla F(x^k)\|^2] \leq \tilde{O}\left(\frac{\kappa^3}{\sqrt{K}} + \frac{\kappa^2 \log K}{\sqrt{K}}\right). \tag{18}$$

## 4 Additional Hyper-Representation and Data Distillation Experiments

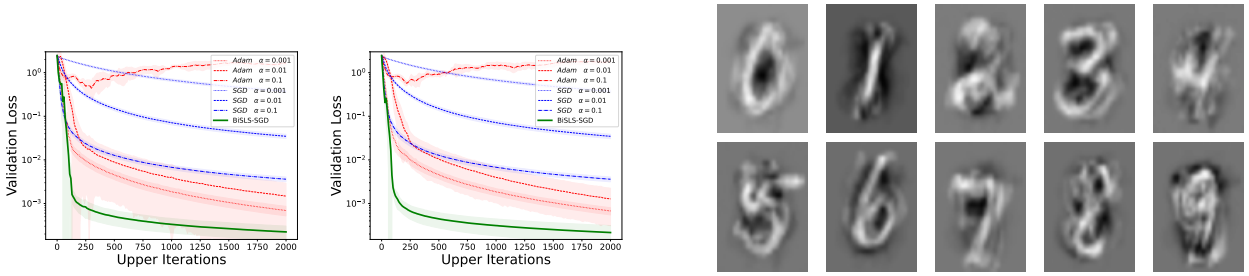
**Hyper-representation learning:** The experiments are performed on MNIST dataset using LeNet [24, 39]. We use conjugate gradient method for solving system of equations when computing the hypergradient [15]. The upper and lower-level objectives are to optimize the embedding layers and the classifier (i.e. the last layer of the neural net), respectively (see Appendix A for details). For constant-step SGD and Adam, we tune the lower-level learning rate $\beta \in \{10.0, 5.0, 1.0, 0.5, 0.1, 0.05, 0.01\}$. For the upper-level learning rate, we tune $\alpha \in \{0.001, 0.0025, 0.005, 0.01, 0.05, 0.1\}$ for SGD, and $\alpha \in \{10^{-5}, 5\cdot 10^{-5}, 10^{-4}, 5\cdot 10^{-4}, 0.001, 0.01\}$ for Adam (recall that $\delta$ in (14) is set to 0). Based on the results of Figure 8, we make the following key observations: ① **line-search at the upper-level is essential for achieving the optimal performance (Figure 8a);** ② **BiSLS-Adam/SGD not only converges fast but also generalizes well (Figure 8b);** ③ **BiSLS-Adam/SGD is highly robust to search starting points** $\alpha_{b,0}$ **and** $\beta_{b,0}$ **(Figure 8c, 8d). It addresses the fundamental question of how to tune** $\alpha$ **and** $\beta$ **in bi-level optimization** (see Appendix A for additional results on search cost).

**Data distillation:** The goal of data distillation is to generate a small set of synthetic data from an original dataset that preserves the performance of a model when trained using the generated data [43, 47]. We adapted the experiment set up from Lorraine et al. [28] to distill MNIST digits. We present the results in Figure 9, where we observe that BiSLS-SGD converges significantly faster than fine-tuned Adam or SGD, and generate realistic MNIST images (see Appendix A for more results).




(a) Validation Loss (b) Validation Accuracy (c) Different $\alpha_{b,0}$ (d) Different $\beta_{b,0}$

Figure 8: Validation loss (a) and accuracy (b) against iterations. (a) Comparisons between whether to use or not use line-search at the upper or lower level; (b) Generalization performance of BiSLS-Adam/SGD and fine-tuned Adam/SGD; (c) Validation loss against iterations for different values of $\alpha_{b,0}$ ($\beta_{b,0}$ fixed at 100). (d) Same plot as (c) but for different values of $\beta_{b,0}$ ($\alpha_{b,0}$ fixed at 10).




(a) Neumann (b) Identity (c) Distilled MNIST Images

Figure 9: (a)(b): Comparison between BiSLS-SGD and Adam/SGD for Data Distillation on MNIST dataset. Validation loss plotted against iterations. (a) Hypergradient computed using Neumann series; (b) Inverse Hessian in (2) treated as the Identity [28] when computing the hypergradient; (c) Distilled MNIST images after 3000 iterations of BiSLS-SGD.

## 5 Conclusion

In this work, we have given simple alternatives to SLS and SPS that show good empirical performance in non-interpolating scenario without requiring the step size to be monotonic. We unify their analysis based on a simplified envelope-type step size, and extend the analysis to the bi-level setting while designing a SPS-based bi-level algorithm. In the end, we propose bi-level line-search algorithm BiSLS-Adam/SGD that is empirically truly robust and adaptive to learning rate initialization. Our work opens several possible future directions. Given the superior performance of BiSLS, we prioritize an analysis of its convergence rates. The difficulty stems from: (a) the bias in hypergradient estimation; (b) the dual updates in $x$ and $y^*(x)$ (incurring nested loop structures); (c) the error in estimating $y^*(x)$. On single-level optimization, we remark as an important direction to relax the two-sample assumption on SPSB /SLSB . On the other hand, we also deem interesting that the two-sample variants of our algorithms do not appear to deteriorate performance in practice. we hope our ideas motivate further research on practical bi-level optimization algorithms that avoid the often computationally exhausting need for tuning step-sizes. At the same time, we hope to see more results alike extending the SPS/SLS beyond the originally investigated setting of interpolating single-level optimization.

**Acknowledgement** This work was partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grants RGPIN-2021-03677 and GPIN-2022-03669.

## References

[1] Zalán Borsos, Mojmir Mutny, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *Advances in Neural Information Processing Systems*, 33:14879–14890, 2020.

[2] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[3] Tianyi Chen, Yuejiao Sun, and Wotao Yin. Tighter analysis of alternating stochastic gradient method for stochastic nested problems, 2021.

[4] Tianyi Chen, Yuejiao Sun, Quan Xiao, and Wotao Yin. A single-timescale method for stochastic bilevel optimization, 2022.

[5] Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex sgd, 2020.

[6] Mathieu Dagréou, Pierre Ablin, Samuel Vaiter, and Thomas Moreau. A framework for bilevel optimization that enables stochastic and global variance reduction algorithms, 2022.

[7] Mathieu Dagréou, Thomas Moreau, Samuel Vaiter, and Pierre Ablin. A lower bound and a near-optimal algorithm for bilevel empirical risk minimization, 2023.

[8] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives, 2014.

[9] James E Falk and Jiming Liu. On bilevel programming, part i: general nonlinear cases. *Mathematical Programming*, 70:47–72, 1995.

[10] Chen Fan, Parikshit Ram, and Sijia Liu. Sign-maml: Efficient model-agnostic meta-learning by signsgd, 2021.

[11] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path integrated differential estimator, 2018.

[12] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

[13] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization, 2017.

[14] Saeed Ghadimi and Mengdi Wang. Approximation methods for bilevel programming, 2018.

[15] Riccardo Grazzi, Luca Franceschi, Massimiliano Pontil, and Saverio Salzo. On the iteration complexity of hypergradient computation, 2020.

[16] Riccardo Grazzi, Massimiliano Pontil, and Saverio Salzo. Convergence properties of stochastic hypergradients, 2021.

[17] Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic, 2022.

[18] Feihu Huang, Junyi Li, and Shangqian Gao. Biadam: Fast adaptive bilevel optimization methods, 2023.

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

[20] Kaiyi Ji, Junjie Yang, and Yingbin Liang. Bilevel optimization: Convergence analysis and enhanced design, 2021.

[21] Kaiyi Ji, Mingrui Liu, Yingbin Liang, and Lei Ying. Will bilevel optimizers benefit from loops, 2022.

[22] Prashant Khanduri, Siliang Zeng, Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. A near-optimal algorithm for stochastic bilevel optimization via double-momentum, 2021.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[25] Xiaoyu Li and Francesco Orabona. On the convergence of stochastic gradient descent with adaptive stepsizes, 2019.

[26] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search, 2019.

[27] Nicolas Loizou, Sharan Vaswani, Issam Laradji, and Simon Lacoste-Julien. Stochastic polyak step-size for sgd: An adaptive learning rate for fast convergence, 2021.

[28] 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.

[29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[30] Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate, 2019.

[31] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

[32] Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *International Conference on Machine Learning*, pages 2613–2621. PMLR, 2017.

[33] Antonio Orvieto, Simon Lacoste-Julien, and Nicolas Loizou. Dynamics of sgd with stochastic polyak stepsizes: Truly adaptive variants and convergence to exact solution, 2022.

[34] Aravind Rajeswaran, Chelsea Finn, Sham Kakade, and Sergey Levine. Meta-learning with implicit gradients, 2019.

[35] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[36] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021.

[37] 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.

[38] Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th international conference on Machine learning*, pages 807–814, 2007.

[39] Daouda Sow, Kaiyi Ji, and Yingbin Liang. On the convergence theory for hessian-free bilevel algorithms, 2022.

[40] Sharan Vaswani, Issam Laradji, Frederik Kunstner, Si Yi Meng, Mark Schmidt, and Simon Lacoste-Julien. Adaptive gradient methods converge faster with over-parameterization (but you should do a line-search). *arXiv preprint arXiv:2006.06835*, 2020.

[41] Sharan Vaswani, Aaron Mishkin, Issam Laradji, Mark Schmidt, Gauthier Gidel, and Simon Lacoste-Julien. Painless stochastic gradient: Interpolation, line-search, and convergence rates, 2021.

[42] Luis Vicente, Gilles Savard, and Joaquim Júdice. Descent approaches for quadratic bilevel programming. *Journal of optimization theory and applications*, 81(2):379–399, 1994.

[43] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. Dataset distillation, 2020.

[44] Rachel Ward, Xiaoxia Wu, and Leon Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes, 2021.

[45] Jorge Nocedal Stephen J Wright. Numerical optimization, 2006.

[46] Junjie Yang, Kaiyi Ji, and Yingbin Liang. Provably faster algorithms for bilevel optimization, 2021.

[47] Ruonan Yu, Songhua Liu, and Xinchao Wang. Dataset distillation: A comprehensive review, 2023.

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

[49] Yihua Zhang, Yuguang Yao, Parikshit Ram, Pu Zhao, Tianlong Chen, Mingyi Hong, Yanzhi Wang, and Sijia Liu. Advancing model pruning via bi-level optimization, 2023.

[50] Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. *arXiv preprint arXiv:2206.00719*, 2022.

# A Additional Experiment Results

This section is organized as follows. First, we discuss synthetic quadratics experiments. Second, we provide more details on the sensitivity of the algorithms to the choices of $\delta$ in (14), on the reset procedure, and on the search cost of BiSLS. Third, we compare the empirical performance of 1-sample vs 2-samples implementations of our algorithms for single-level convex and bi-level optimization. Some additional results for hyper-representation learning and data distillation experiments are also presented. We run 5 independent runs for all our experiments.

## A.1 Synthetic Quadratics

The experiments on quadratic functions are adapted from Loizou et al. [27]. The training objective is as follows:

$$f(x) = \frac{1}{2}(x - x_1^*)^T H_1 (x - x_1^*) + \frac{1}{2}(x - x_2^*)^T H_2 (x - x_2^*),$$

where $H_i$ $(i = 1, 2)$ are positive definite. The optimal solutions $x_i^*$ $(i = 1, 2)$ are generated randomly from a standard normal distribution. Specifically, $H_i$ is defined as follows:

$$H_i = O^T \cdot \text{Diag}(\log(1 + \lambda_i)) \cdot O, \quad i = 1, 2,$$

where $O$ and $\lambda_i$ are taken from the spectral decomposition of $P^T P$, and $P$ is generated from the standard normal distribution. Figure 10 shows the convergence of various algorithms with different starting points. Interestingly, both SPSB with either 1 sample or 2 samples (1 sample for computing the gradient and the other for computing the step size) converge to the optimal solution (labelled with a star).

### A.2 Sensitivity of $\delta$, reset, and search cost

In this section, we discuss the effects of $\delta$ in (14), $\eta$ in Algorithm 3, and comment on the search cost of the options in the Reset Algorithm 3. Recall that the line-search condition (8) assumes that we can find a largest $\gamma_k \leq \gamma_{b,0}$ to satisfy it. However, in practice, we apply a backtracking procedure, i.e. $\gamma_k = \gamma_k * w, 0 < w < 1$, until $\gamma_k$ satisfies (8). Therefore, the found learning rate is not guaranteed to be the largest. Nonetheless, we assume that $\gamma_k$ is the largest to simplify our analysis given above (similar arguments apply to line-search at both upper and lower-level in the bi-level optimization). The experiments in this section are based on hyper-representation learning [39]. In this case, the objective of the induced bilevel-optimization problem can be written as:

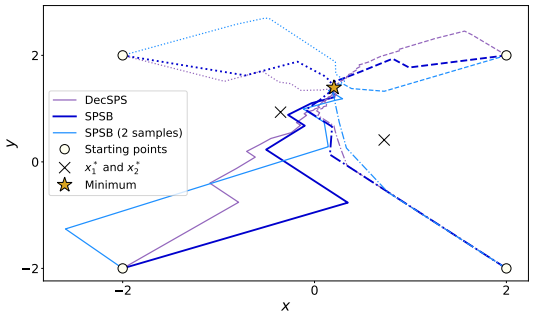


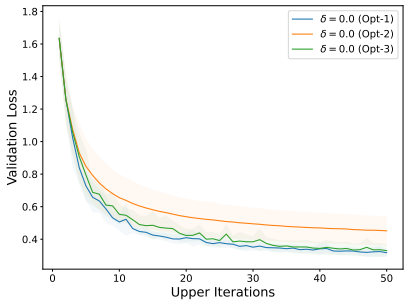

Figure 10: Iterate trajectories of different starting points for the synthetic quadratic experiments.

$$\min_w F(w) = \frac{1}{2D_{X_1}} \|\tilde{f}(X_1; w)c^*(w) - Y_1\|^2$$
$$s.t. \quad c^*(w) = \operatorname*{argmin}_c \frac{1}{2D_{X_2}} \|\tilde{f}(X_2; w)c - Y_2\|^2 + \frac{\lambda}{2}\|c\|^2,$$

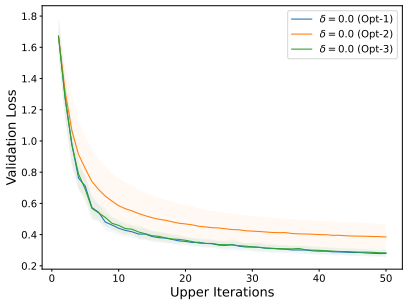
where $(X_1, Y_1)$ and $(X_2, Y_2)$ are validation and training data sets with sizes $D_{X_1}$ and $D_{X_2}$, respectively; $\tilde{f}(\cdot; w)$ are the embedding layers of the model parameterized by $w$; and, $c$ is the classification layer. Moreover, we use conjugate gradient methods (CG) [15, 16] to solve the linear system when computing the hypergradient for hyper-representation learning experiments.

**Reset** While Algorithm 3 (reset) can be applied to both upper and lower-level problems, we focus our discussions here on the upper-level learning rate $(\alpha_k)$. This is because we empirically find it to be more critical for the convergence performance (see Figure 8a). As shown in Algorithm 3, reset has 3 options. Options 1, 2, and 3 search starting from $\alpha_{b,0}$, $\alpha_{k-1}$, and $\eta\alpha_{k-1}$, at iteration $k$ respectively. Option 1 has the highest search cost as it always starts from the same initial upper bound $(\alpha_{b,0})$. Option 2 ensures the monotonicity of the learning rate due to $\alpha_k \leq \alpha_{b,k} = \alpha_{k-1}$. Option 3 chooses the search starting point at iteration $k$ $(\alpha_{b,k})$ by multiplying the previous learning rate $(\alpha_{k-1})$ by a factor $\eta \geq 1$. As in the single-level convex case where monotonicity in the step size can potentially lead to slow convergence (see Figure 5), we again observe that monotonicity in the upper learning rate (i.e. option 2) leads to poorer performance when compared against options 1 or 3 as shown in Figure 11. Finally, we compare the performance of different $\eta$s in option 3 (note that $\eta = 1$ in option 3 is equivalent to option 2). We observe in Figure 12 that different $\eta$s perform equally well. This shows the robustness of our algorithm to the choice of $\eta$. As mentioned previously, the choice of option 3 over option 1 are due to 2 reasons: (a) reduced search cost; (b) provides an overall non-increasing and non-monotonic trend of upper bound $\alpha_{b,k}$. We discuss search cost of different $\eta$s in option 3 below.

**Search Cost** We investigate the line-search cost based on options 1, 2, and 3 in reset (Algorithm 3) for the upper-level problem. For option 1, the search cost in terms of number of search rounds (i.e. number of evaluations using (14)) per iteration for BiSLS-SGD and BiSLS-Adam is $89 \pm 15$ and $115 \pm 16$, respectively. The search cost of BiSLS-Adam is higher than that of BiSLS-SGD because the feasible learning rate range for Adam is typically smaller than SGD at the upper-level. Based on the results in Figure 13, we observe the use of option 3 in reset can significantly reduce the search cost for both BiSLS-Adam and BiSLS-SGD. For example, choosing $\eta = 2$ in option 3 results in an average upper-level search cost of only $\sim 9$ rounds per iteration, which is much smaller than that of option 1. As we have shown in Figure 11, options 1 and 3 have nearly the same performance. Therefore, option 3 is an efficient algorithm that maintains good performance while reducing computation cost. Option 2 has the lowest search cost ($\sim 4$ rounds per iteration). However, its performance is not as good as option 1 or 3 as observed in Figure 11. Moreover, the average lower-level search cost is only 1 round per iteration when option 1 is used (see Figure 13b).

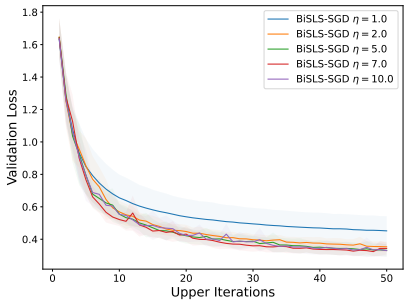



(a) Different Opt (BiSLS-SGD)

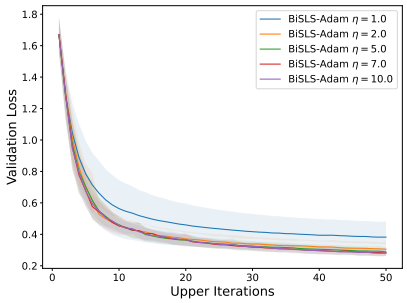



(b) Different Opt (BiSLS-Adam)

Figure 11: Validation loss against iterations with search options 1, 2, and 3 for the upper-level learning rate. The results for BiSLS-SGD and BiSLS-Adam are in (a) and (b), respectively. For the lower-level search, we fix it option 1 with $\beta_{b,0} = 100$. Results are based on hyper-representation learning.




(a) Different $\eta$s (BiSLS-SGD)




(b) Different $\eta$s (BiSLS-Adam)

Figure 12: Validation loss against iterations for different $\eta$s based on reset option 3. Results for BiSLS-SGD are given in (a) and for BiSLS-Adam are given in (b). Note that $\eta = 1$ in reset option 3 is equivalent to reset option 2. For the lower-level search, we fix it option 1 with $\beta_{b,0} = 100$. Results are based on hyper-representation learning.

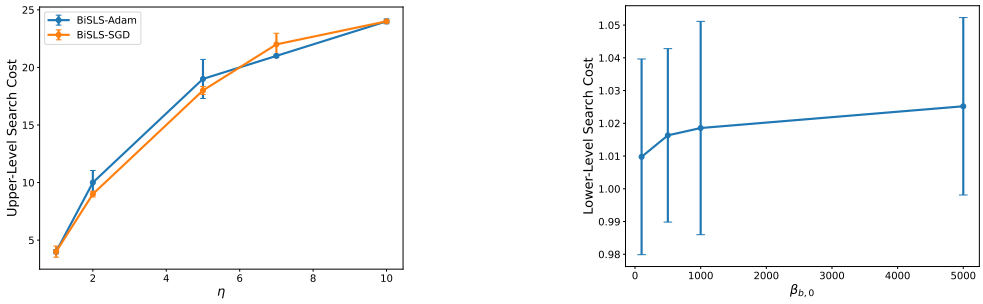
**Sensitivity on $\delta$** As mentioned in Sec 2, due to the stochastic error in hypergradient computation, further complicated by the approximation error of $y^*(x)$ (see (14)), a learning rate is not guaranteed to be found in the bi-level case. Specifically, this is in contrast to the single-level convex problems. To avoid this, we introduce in (14) a $\delta$ slack to give some tolerance to such errors. Here, we give a thorough investigation of the effects of $\delta$ on performance. We vary its magnitude across 6 orders for both reset options 1 and 3 (see Algorithm 3 and discussions on reset above). We observe that despite a large difference on the magnitudes of $\delta$, they all share very similar performance for both BiSLS-SGD and BiSLS-Adam: see Figures 14 and 15. We summarize the key fins in this section as follows: ① **The option 3 in reset has good empirical performance (outperforms option 2) and is an effective way to reduce search cost (Figure 11, 13); ② BiSLS is highly robust to different choices of $\eta$ in option 3 and $\delta$ in** (14) **(Figure 12, 14, 15).**

### A.3 Data distillation objective and additional results

We let $\mathcal{L}_S(w)$ denote the loss evaluated on dataset $S$ with model weights $w$. The objective of *data distillation* can be expressed as a BO problem as follows:

$$D^* = \underset{D}{\operatorname{argmin}} \mathcal{L}_{\tilde{V}}(w^*(D)) \quad \text{s.t.} \quad w^*(D) = \underset{w}{\operatorname{argmin}} \mathcal{L}_D(w),$$

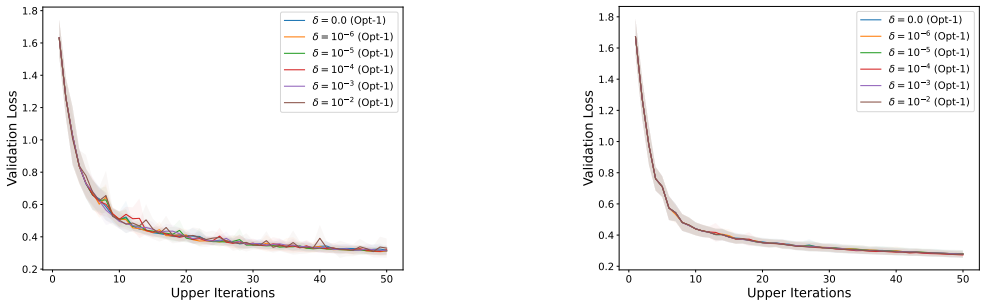
where $\tilde{V}$ is of the same size as $D$ and subsampled from the entire (original) dataset $V$. The solution $D^*$ is the distilled data, e.g. 9 MNIST digits each corresponding to a different label. In figure 16a, we show the performance of BiSPS for different values of $\alpha_{b,0}$ in comparison with BiSLS-SGD, and observe that BiSLS-SGD has better performance. In 16b, we show the results when we increase the number of lower-level iterations (T) from 20 to 50. As observed for $T = 20$ (in Figure 9), BiSLS-SGD here also outperforms a fine-tuned Adam or SGD.




(a) Upper-level search cost

(b) Lower-level search cost

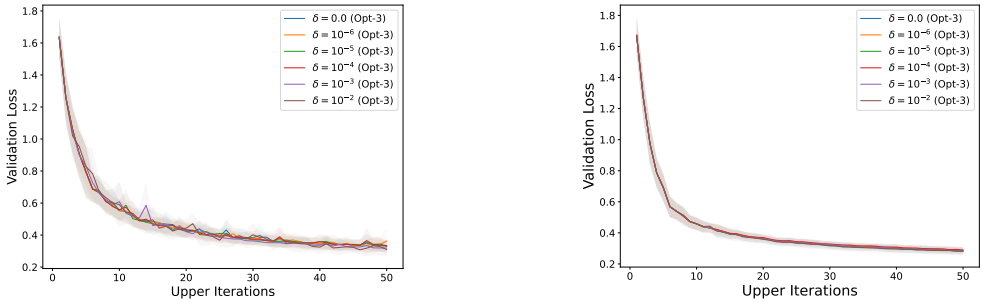
Figure 13: (a) Upper-level search cost measured in terms of average number of search rounds per iteration against different $\eta$s in reset (Algorithm 3). (b) Lower-level search cost measured in the same way as upper-level against different lower-level search starting points ($\beta_{b,0}$). The lower-level search is done with option 1 (see above for discussions about these options).




(a) Different $\delta$s/Opt-1 (BiSLS-SGD)

(b) Different $\delta$s/Opt-1 (BiSLS-Adam)

Figure 14: Validation loss against iterations for different $\delta$s based on reset option 1. Results for BiSLS-SGD (a) and for BiSLS-Adam (b). For the lower-level search, we fix it option 1 with $\beta_{b,0} = 100$. Results are based on hyper-representation learning.



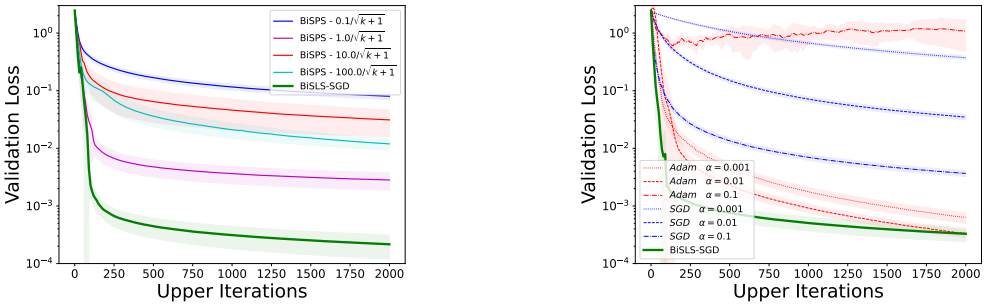

(a) Different $\delta$s/Opt-3 (BiSLS-SGD)

(b) Different $\delta$s/Opt-3 (BiSLS-Adam)

Figure 15: Validation loss against iterations for different $\delta$s based on reset option 3 ($\eta = 10$). Results for BiSLS-SGD (a) and for BiSLS-Adam (b). For the lower-level search, we fix it option 1 with $\beta_{b,0} = 100$. Results are based on hyper-representation learning.

### A.4 1-sample or 2-samples versions of algorithms for convex and bi-level optimization

We provide additional results to compare the performance of 1-sample and 2-samples (one for computing the gradient and the other for computing the step size) versions of our algorithms for SPSB and BiSPS used for single-level and bi-level optimization, respectively. In the single-level case (Figure 17), we observe that 2-samples SPSB performs just as well as 1-sample SPSB . Interestingly, we observe that their step sizes also follow a similar pattern. That is: an initial increase

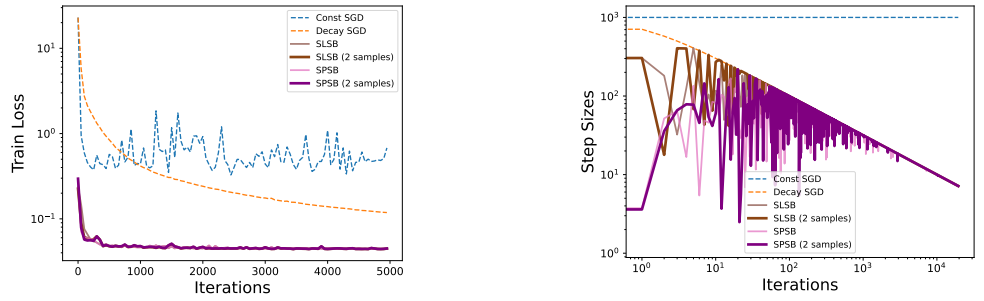



(a) Different $\alpha_{b,0}$s (BiSPS)

(b) T = 50, Identity

Figure 16: Validation loss against iterations. (a) Comparison between BiSPS with different $\alpha_{b,0}$s and BiSLS-SGD. (b) Comparison between BiSLS-SGD to fine-tuned Adam/SGD ($\beta_k$ fixed at $10^{-4}$). Inverse Hessian in (2) treated as the Identity [28] when computing the hypergradient. Recall that T is the total number of lower-level iterations and we have shown the results for $T = 20$ in Figure 9b.

followed by a regime where $\gamma_k = \frac{f_{i_k}(x^k) - l^*_{i_k}}{c\|\nabla f_{i_k}(x^k)\|^2}$ is frequently used, and eventually changes to decaying-step SGD. This seems to also match with Theorem 2 where a transition point for SPSB ($k_0 = \max\{1, \lceil \gamma_0/\omega \rceil - 1\}, w = \frac{1}{2cL_{\max}}, \gamma_0 \geq \frac{1}{\mu}$) is predicted. At the same time, we also note that (perhaps, unsurprisingly) the 2-samples version seems to have a slightly more oscillatory behavior than the 1-sample version as shown in Figure 17. SLSB with either 1-sample or 2-samples also result in a similar performance and step size. Overall, despite the requirements of Theorems 1 and 2 for a 2-samples assumption, the empirical performance of 1-sample and 2-samples for either SPSB or SLSB appears to be very similar. Moving on to the bi-level case, recall that Theorems 3 and **??** require the 2-samples assumption (i.e., $\alpha_k$ independent of $h^k_f$) for the upper-level learning rate. We empirically verify this assumption with both hyper-representation learning and data distillation experiments. For hyper-representation learning experiments in Figure 18, BiSPS with either 1-sample or 2-samples for different values of $\alpha_{b,0}$ show similar performance. In fact, for $\alpha_{b,0} = 0.1$ we even observe that the 2-samples variant outperforms the 1-sample BiSPS. For data distillation experiments in Figure 19, the performances of 1-sample and 2-samples BiSPS are similar to each other when $\alpha_{b,0} = 10.0$ or $\alpha_{b,0} = 50.0$. In general, the performance difference between 1-sample and 2-samples in the single-level or bi-level settings is small.




(a) Train loss

(b) Step size

Figure 17: Binary linear classification on w8a dataset using logistic loss [2]. Train loss (left) and step size (right) against iterations. We choose $\gamma_{b,0} = 1000$ for all algorithms. The upper bound for either SPSB or SLSB decays as $\gamma_{b,k} = \frac{\gamma_{b,0}}{\sqrt{k+1}}$. For decaying-step SGD, the learning rate schedule is $\frac{\gamma_{b,0}}{\sqrt{k+1}}$.

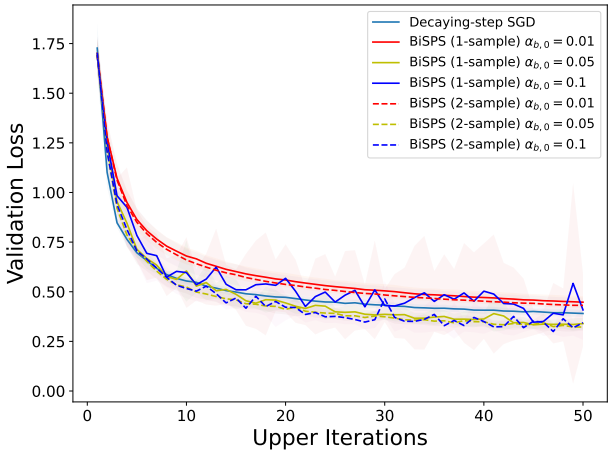



Figure 18: Comparison between BiSPS (2-samples), BiSPS (1-sample) and decaying-step SGD. Experiments are based on hyper-representation learning. For either version of BiSPS, the lower-level learning rate ($\beta_k$) is fixed at 10. The hypergradient is computed using conjugate gradient [15].

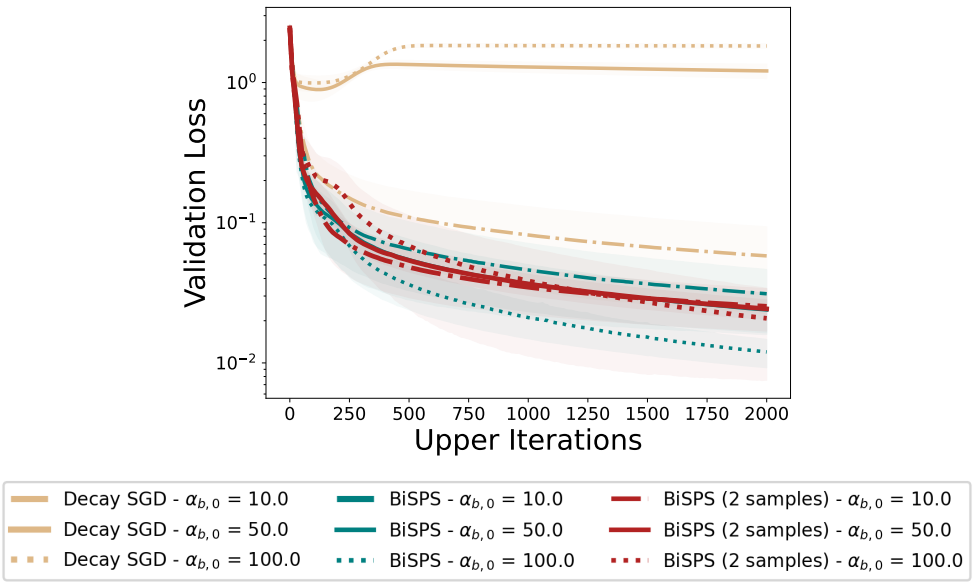



Figure 19: Comparison between BiSPS (2-samples), BiSPS (1-sample) and decaying-step SGD. Experiments are based on data distillation. For either version of BiSPS, the lower-level learning rate ($\beta_k$) is fixed at $10^{-4}$. The Inverse Hessian in (2) is treated as the Identity when computing the hypergradient [28].