Contributions to stochastic bilevel optimization

Mathieu Dagreou

université
PARIS-SACLAY

Contributions to stochastic bilevel optimization

Contributions à l'optimisation bi-niveaux stochastique

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580 : Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat: Informatique mathématique
Graduate School : Informatique et sciences du numérique
Référent : Faculté des sciences d'Orsay

Thèse préparée dans l'unité de recherche **Inria Saclay-Île-de-France** (Université Paris-Saclay, Inria), sous la direction de **Samuel VAITER** (HDR), Chargé de Recherche, le co-encadrement de **Thomas MOREAU**, Chargé de Recherche, et le co-encadrement de **Pierre ABLIN**, Chercheur.

Thèse soutenue à Paris-Saclay, le 08 octobre 2024, par

Mathieu DAGRÉOU

Composition du jury

Membres du jury avec voix délibérative

Édouard PAUWELS Professeur des universités, Toulouse School of Economics	Président
Aurélien BELLET Directeur de recherche, Inria, Université de Montpellier	Rapporteur & Examinateur
Peter OCHS Professeur, Université de Saarland	Rapporteur & Examinateur
Émilie CHOUZENOUX Directrice de recherche, Inria, Université Paris-Saclay	Examinatrice
Julien MAIRAL Directeur de recherche, Inria, Université de Grenoble	Examinateur

THESE DE DOCTORAT
NNT : 2024UPASG041

CONTENTS

REMERCIEMENTS

Je souhaite en premier lieu exprimer toute ma gratitude envers mes directeurs de thèse. Merci de m'avoir fait confiance il y a maintenant pratiquement 3 ans et demi pour mener à bien ce projet. Merci à vous trois de m'avoir fait grandir en tant que chercheur. Je mesure la chance que j'ai eue de travailler avec vous trois, ce qui m'a permis de découvrir tout le spectre entre la théorie et la pratique. Merci à Thomas de m'avoir tant fait progresser en code et de m'avoir impliqué dans Benchopt. Non, je ne t'en veux pas des multiples changements d'API et je suis certain que le benchmark bilevel a encore de beaux jours devant lui ! Merci à Pierre pour ta disponibilité et ta pédagogie. Je garde en mémoire ces séances au tableau à essayer de débugger ces multiples inégalités. Merci à Samuel pour ta bienveillance et les multiples séjours à Nice. Même si l'éloignement géographique fait que l'on a moins interagi, les discussions que l'on a pu avoir tant sur le plan scientifique que sur le monde académique m'ont été très précieuses ! Ce qui est sûr, c'est que les meetings du vendredi vont me manquer et que j'espère que nous aurons l'occasion de collaborer dans le futur !

I thank Aurélien and Peter for reviewing this document and providing valuable comments. I know that July and August are not the best months to review a PhD manuscript since you needed to do it between summer break and the NeurIPS reviewing process. I am therefore grateful for the time you spent on it. I also thank the other members of the jury Émilie, Julien, and Édouard for attending the defense and for their insightful questions.

La fin de cette thèse marque également la fin de mon appartenance à l'équipe MIND/SODA, anciennement PARIETAL. Je souhaite remercier tous les stagiaires, doctorants et postdoctorants qui ont permis de rendre plus agréables ces trois années. Merci à Benoît et Cédric pour les discussions du lundi matin à Turing, mais ausi pour le séjour à Pise ! Merci à ceux qui ont partagé le bureau 1047 avec moi : Bénédicte, Fatemeh, Houssam, Judith et Marine. Je remercie également tous les autres pour les différents afterworks, l'ambiance au quotidien ainsi que lors des retraites d'équipe et les différents voyages en conférence ou en sprint Benchopt : Alexandre[4], Alexis, Ambroise, Antoine, Apolline, Binh, Charlotte, Cédric, Célestin, Félix, Florent, Guillaume, Hugo, Jad, Jade, Joseph, Jovan, Julia, Julie, Julien, Léo, Lilian, Louis[2], Marie, Mansour, Matthieu[2], Merlin, Nicolas, Omar, Pierre-Antoine, Pierre-Louis, Raphaël, Riccardo, Samuel[2], Sébastien, Théo, Thomas, Virginie, Zaccharie, et tous ceux que j'oublie. Merci à tous les PIs des équipes MND/SODA/PARIETAL passés et présents pour réussir à maintenir un environnement scientifique stimulant et continuer à se battre pour des conditions de travail exceptionnelles : Alexandre, Bertrand, Chaithya, Demian, Gaël, Jill-Jênn, Judith, Marine, Philippe et Thomas.

Je tiens à remercier Gabriel Peyré de m'avoir accueilli tous les vendredis au sein des bureaux du CSD à l'École Normale Supérieure. Au-delà de l'expérience gustative des vendredis midis, il a été très agréable de côtoyer un environnement scientifique différent de l'Inria. Je remercie tous les membres du CSD que j'ai croisé toutes les semaines : Francisco, Geert, Jules, Kimia, Michael, Othmane, Raphaël,

Sibylle, Thibault, Valérie et tous les autres.

Je n'oublie pas toutes les personnes que j'ai pu croiser lors des conférences, workshops et sprints Benchopt que ce soit à la Nouvelle-Orléans, à Valence, à Toulouse, à Arcachon, à Montpellier, à Nancy ou à Grenoble. Je pense en particulier à Benjamin, Quentin, Hugues, Joseph, Mathurin, Paul, Rémy, Ryan, Tam, Théo, Yanis entre autres.

Je souhaite avoir une pensée les amis que je côtoie sur et à côté des pistes d'escrime depuis tant d'années que ce soit les week-ends en compétition ou certains soirs à Paris Nord.

Enfin, je remercie mes parents, ma sœur, mon frère ainsi que le reste de ma famille pour leur soutien tout au long de ces années.

NOTATIONS

General

$\|\cdot\|$	Scalar product
$\langle\cdot,\cdot\rangle$	Euclidean norm
$X^\dagger$	The pseudoinverse of X
$\mathrm{sign}(x)$	Sign function, $\mathrm{sign}(x)=1$ if $x>0$, $\mathrm{sign}(x)=-1$ if $x<0$ and $\mathrm{sign}(0)=0$

Sets

$\mathbb{N}$	Set of non-negative numbers
$\mathbb{N}_{>0}$	Set of positive integers
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^n$	Set of real valued vectors of dimension n
$\mathbb{R}^{m\times n}$	Set of real valued matrices of dimension $m\times n$
$[n]$	The set of integers $\{1,\dots,n\}$
$\mathcal{Y}^\mathcal{X}$	The set of functions defined of $\mathcal{X}$ and taking values in $\mathcal{Y}$

Differential operators

$\nabla f(x)$	Gradient of f at x
$\nabla_y f(x,y)$	Partial gradient of f with respect to the variable y at (x,y)
$\mathrm{d}f(x)$	Differential of f or Jacobian of f at x
$\partial_y f(x,y)$	Partial differential of f or partial Jacobian of f with respect to y at (x,y)
$\nabla^2 f(x)$	Hessian of f at x

Landau notation

$f(x)=\mathcal{O}_a(g(x))$	There exists a constant $C>0$ independent from x and a neighborhood $\mathcal{V}_a$ of a such that $\|f(x)\|\leq C\|g(x)\|$ for all $x\in\mathcal{V}_a$. When the context is clear, we will omit the subscript a.

$f(x) = \tilde{\mathcal{O}}_a(g(x))$	There exists a constant $C > 0$ independent from x and a neighborhood $\mathcal{V}_a$ of a such that $\|f(x)\| \leq C \mid \log(x^{-1})g(x) \mid$ for all $x \in \mathcal{V}_a$. When the context is clear, we will omit the subscript a.
$f(x) = \Omega_a(g(x))$	There exists a constant $C > 0$ independent from x and a neighborhood $\mathcal{V}_a$ of a such that $C\|g(x)\| \leq \|f(x)\|$ for all $x \in \mathcal{V}_a$. When the context is clear, we will omit the subscript a.
$f(x) = \Theta_a(g(x))$	There exists constant $C_1 > 0$ and $C_2 > 0$ independent from x and a neighborhood $\mathcal{V}_a$ of a such that $C_1\|g(x)\| \leq \|f(x)\| \leq C_2\|g(x)\|$ for all $x \in \mathcal{V}_a$. When the context is clear, we will omit the subscript a.

CHAPTER 1

INTRODUCTION

Sommaire

The recent years have witnessed the development of cutting-edge technologies based on artificial intelligence across various fields such as healthcare, finance, and social science. These advancements, driven by machine learning, have been enabled by the combination of vast data availability, increased computational power through the development of GPUs, and the creation of increasingly complex and expressive models. Machine learning algorithms enable systems to automatically learn tasks by observing data examples and generalizing from them. This data can take many forms, including images, tabular data, and text. Typical tasks include generating new data samples (e.g., generating realistic images), classification (e.g., predicting whether a picture depicts a dog or a cat), and regression (e.g. predicting a house's price based on features like size or location).

To perform these tasks, we generally consider a parametric model (e.g., a neural network) and a measure of error, which quantifies how well the model performs on data. Training the model involves finding the model's parameters that minimize this error measure using an optimization algorithm. This approach has led to the development of efficient optimization algorithms tailored to the specific structure of machine learning problems.

Recently, machine learning problems with a hierarchical structure have emerged. In these problems, evaluating a potential solution requires solving an auxiliary problem that depends on the considered

solution. For instance, when evaluating a model's generalization performance, we train it on training samples and then estimate its performance by computing the error on validation samples. Bilevel optimization provides a natural framework for formalizing such hierarchical problem structures. Due to its wide range of applications, such as hyperparameter selection, training deep equilibrium models, and meta-learning, bilevel optimization has gained significant interest in the machine learning community. This interest has led to the development of new algorithms explicitly designed for bilevel structures. While these algorithms share some similarities with classical single-level optimization algorithms, efforts are needed to enhance their efficiency and understand their behavior.

In this chapter, we introduce the formalism of bilevel optimization and demonstrate its usefulness for various machine learning applications.

1.1 Introducing example: from hyperparameter selection to bilevel optimization

This section details how one can translate a machine learning problem into a bilevel optimization problem. We take the hyperparameter selection problem as an example.

1.1.1 Supervised learning

In what follows, we briefly overview the supervised learning paradigm. For more detailed presentations, interested readers can refer to the following textbooks: (Hastie et al., 2009; Shalev-Shwartz and Ben-David, 2014; Bach, 2021).

Supervised learning is a machine learning paradigm aimed at predicting an outcome $y \in \mathcal{Y}$ from an input $x \in \mathcal{X}$. The input x can be an image, for example. The nature of the output y depends on the task: either regression or classification. In regression, the output y is quantitative and takes values in a continuous space, such as $\mathcal{Y} = \mathbb{R}$ (e.g., temperature or house price). In classification, the output y is categorical and takes values in a finite set. For instance, in binary classification, $\mathcal{Y} = \{-1\,,\,1\}$ (e.g., -1 if the input image does not contain a cat, 1 if it contains a cat), and in multi-class classification, $\mathcal{Y} = \{\,1\,,\,\ldots\,,\,K\,\}$.

Mathematically, pairs of input/output (x, y) can be viewed as realizations of a random variable (X, Y), which takes values in the measurable space $\mathcal{X} \times \mathcal{Y}$, endowed with a joint distribution $\mathbb{P}_{(X,Y)}$. Predictions are performed using a prediction function h defined on $\mathcal{X}$. Therefore, learning to predict Y from X involves finding the best prediction function h from a hypothesis set $\mathcal{H} \subset \mathcal{Y}^{\mathcal{X}}$ such that $h(X) \approx Y$.

To evaluate the quality of a prediction, we introduce a loss function $\ell : \mathcal{Y}^2 \to \mathbb{R}^+$ that measures the discrepancy between two elements of $\mathcal{Y}$. This loss function is such that $\ell(y, y')$ is small if y and y' are close and large otherwise.

Example 1.1. *These are classical choices of loss function depending on the downstream task:*

- *The squared loss defined by* $\ell(y, y') = \frac{1}{2}(y - y')^2$ *is often used in regression tasks where* $\mathcal{Y} = \mathbb{R}$.
- *The binary loss defined by* $\ell(y, y') = \mathbb{1}_{y \neq y'}$ *is the most natural loss for classification tasks where* $\mathcal{Y} = \{-1, 1\}$ *or* $\mathcal{Y} = \{1, \ldots, K\}$.

The previous binary loss poses computational challenges because it is non-convex and non-differentiable. This makes the problem of learning directly with this loss NP-hard (Ben-David et al., 2003; Feldman et al., 2012). In practice, convex surrogates are preferred because they are easier to optimize.

- *The logistic loss* $\ell(y, y') = \log(1 + \mathrm{e}^{-yy'})$ *is often used in binary classification. For this loss, the constraint of having* $y' \in \{-1, 1\}$ *is relaxed to* $y' \in \mathbb{R}$.

- *The logistic loss can be extended to multi-class regression loss. In this case, if we have K classes, we can encode the output y as a one-hot vector $y \in \{0,1\}^K$ and the prediction y' as a vector in $[0,1]^K$. Then, the loss considered is the cross-entropy loss given by $\ell(y,y') = -\sum_{k=1}^K y_k \log(y'_k)$.*

To get a measure of performance of a prediction function $h \in \mathcal{H}$ on the joint distribution $\mathbb{P}_{(X,Y)}$, we introduce the expected risk is defined by

$$\mathcal{R}(h) = \mathbb{E}_{(X,Y)\sim\mathcal{P}_{(X,Y)}}\left[\ell(Y,h(X))\right] \ . \tag{1.1}$$

The lower $\mathcal{R}(h)$, the better the prediction function h. Thus, one can find the best prediction function by minimizing the expected risk over the set of functions $\mathcal{H}$. This yields the following optimization problem

$$\min_{h\in\mathcal{H}} \mathcal{R}(h) \ . \tag{1.2}$$

The function h is often chosen among a parametric family of functions $(h_\theta)_{\theta\in\Theta}$ where Θ is a subset of $\mathbb{R}^p$. This converts Problem (1.2) into the following finite dimensional optimization problem

$$\min_{\theta\in\Theta} \mathcal{R}(h_\theta) \ . \tag{1.3}$$

In practice, we do not have access to the joint distribution $\mathbb{P}_{(X,Y)}$, making the computation of $\mathcal{R}(h_\theta)$ intractable. We would rather have access to a finite number of realizations of (X,Y). We denote $\mathcal{D}^{\text{train}} = \{(x_i^{\text{train}}, y_i^{\text{train}})\}_{i=1}^n$ these realizations. The set $\mathcal{D}^{\text{train}}$ is called the training set; its elements are the training samples. The *Empirical Risk Minimization* (ERM) problem consists in using the empirical average of the loss over the training set as a proxy for the expected risk, leading to Problem (1.4)

$$\min_{\theta\in\Theta} \hat{\mathcal{R}}(\theta) \triangleq \frac{1}{n}\sum_{i=1}^n \ell(y_i^{\text{train}}, h_\theta(x_i^{\text{train}})) \ . \tag{1.4}$$

Example 1.2. *The most simple instance of this problem is the least squares regression problem. It consists in assuming that the following linear model links Y and X*

$$Y = \langle X, \theta^*\rangle + \sigma\varepsilon$$

with $\varepsilon \sim \mathcal{N}(0,1)$, $\sigma > 0$ and some $\theta^* \in \mathbb{R}^p$ to estimate. If $(x_1^{\text{train}}, y_1^{\text{train}}), \dots, (x_n^{\text{train}}, y_n^{\text{train}})$ are i.i.d. realizations of (X,Y), we can estimate θ by maximizing the likelihood of the training samples

$$\max_{\theta\in\mathbb{R}^p} \prod_{i=1}^n p(y_i^{\text{train}} | x_i^{\text{train}}, \theta) = \exp\left[\frac{1}{2\sigma^2}\sum_{i=1}^n (y_i^{\text{train}} - \langle x_i^{\text{train}}, \theta\rangle)^2\right] \ .$$

By taking the logarithm, adding a minus sign, and removing constants in the previous expression, finding the Maximum Likelihood Estimator of θ^* is equivalent to solving the following ERM problem

$$\min_{\theta\in\mathbb{R}^p} g(\theta) = \frac{1}{2n}\sum_{i=1}^n (\langle x_i^{\text{train}}, \theta\rangle - y_i^{\text{train}})^2 \ .$$

With the notation of Problem (1.4), the prediction function h_θ is defined by $h_\theta(x) = \langle x, \theta\rangle$ and the loss function ℓ is the squared loss $\ell(y,y') = \frac{1}{2}(y-y')^2$.

Let us denote the design matrix $\mathbf{X} = \begin{bmatrix} x_1^{\text{train}} & \dots & x_n^{\text{train}} \end{bmatrix}^\top \in \mathbb{R}^{n\times p}$ and $\mathbf{Y} = (y_1^{\text{train}}, \dots, y_n^{\text{train}}) \in \mathbb{R}^n$. The function g can be written as

$$g(\theta) = \frac{1}{2n}\|\mathbf{X}\theta - \mathbf{Y}\|^2 \ .$$

The function g being convex and differentiable over $\mathbb{R}^p$, we will see in Chapter 2 that minimizing it is equivalent to finding the zeros of its gradient. Since the gradient of g is given by $\nabla g(\theta) = \frac{1}{n}\mathbf{X}^\top(\mathbf{X}\theta - \mathbf{Y})$, a least squares estimator is a solution of the following linear system

$$\mathbf{X}^\top\mathbf{X}\theta = \mathbf{X}^\top\mathbf{Y} \ . \tag{1.5}$$

The solutions of Equation (1.5) are given by the affine space $\ker(\mathbf{X}) + \mathbf{X}^\dagger\mathbf{Y}$. In particular, if $\mathbf{X}$ is full rank, the least squares estimator is unique and is given by $\theta^* = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}$.

1.1.2 Generalization and regularization

Assume that $\hat{\mathcal{R}}$ defined in Problem (1.4) has a minimizer θ^* over the parameter space Θ. This ensures that the prediction function h_{θ^*} is the one that performs best on the training samples among all the $(h_\theta)_{\theta\in\Theta}$. However, our goal is not just to perform well on the training set but also to predict accurately for any new input-output pair (x, y) drawn from the joint distribution $\mathbb{P}_{(X,Y)}$. This is known as the notion of generalization. A possibility to measure the generalization performance of h_{θ^*} is to use a validation set $\mathcal{D}^{\text{val}} = \{(x_j^{\text{val}}, y_j^{\text{val}})\}_{j=1}^m$ distinct from the training set and evaluate the average error on this validation set, often referred to as the hold-out loss

$$f(\theta^*) \triangleq \frac{1}{m}\sum_{j=1}^{m} \ell(y_j^{\text{val}}, h_{\theta^*}(x_j^{\text{val}})) \ . \tag{1.6}$$

A high value of $f(\theta^*)$ coupled with a low value of the training error $g(\theta^*)$ indicates that the model fits the training data too closely but does not generalize well to new data. This phenomenon is known as overfitting. Overfitting is closely related to the complexity of the model, which refers to the model's capacity to approximate complex functions. Generally, more complex models require more training samples to avoid overfitting.

Example 1.3. *To illustrate the overfitting phenomenon, we consider a polynomial regression problem. We generate realizations of (X, Y) as follows: $X \sim \mathcal{U}([-1, 1])$ and $Y = f(X) + \sigma\varepsilon$ with f defined by $f(x) = x^2 - \frac{1}{2}$, $\varepsilon \sim \mathcal{N}(0, 1)$ and $\sigma = 0.1$. For $p \in \{1, \dots, 15\}$ and $\theta \in \mathbb{R}^p$, we take as prediction function, a polynomial of degree $p - 1$*

$$h_\theta(x) = \sum_{k=0}^{p-1} \theta_k x^k \ .$$

In this setting, the higher the degree of the polynomial, the more complex the model. We learn a vector $\theta \in \mathbb{R}^p$ by solving the following least squares problem

$$\min_{\theta\in\mathbb{R}^p} g(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i^{\text{train}} - h_\theta(x_i^{\text{train}}))^2 \ . \tag{1.7}$$

In figure 1.1, we display the training samples, and we plot the learned prediction function h_{θ^*} for the different values of p and the function f. We observe an underfitting phenomenon for the degrees 0 and 1: the training error is high because linear models are not expressive enough. We observe an overfitting phenomenon for the highest degrees: the prediction function fits very well on the training samples but does not generalize well. This is confirmed by figure 1.2, which shows that the training error decreases with the degree while the test error decreases and then increases.

Overfitting can be prevented by limiting the model's complexity. One approach is to reduce the size of the set of prediction functions under consideration. For instance, in Example 1.3, this involves lowering the polynomial degree of the prediction function. Another method is to regularize the optimization problem as shown in Problem (1.4). It consists in adding a penalty function $\Omega : \Theta \times \Lambda \to \mathbb{R}^+$ to the empirical risk which helps control the model's complexity. The regularization term is parametrized by

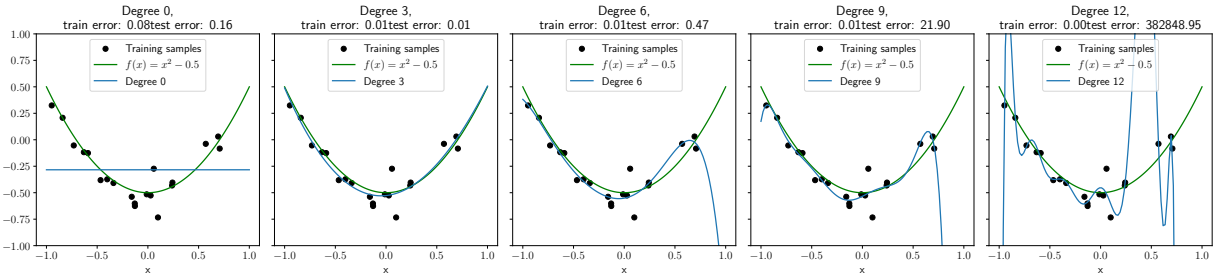


Figure 1.1: Illustration of overfitting in polynomial regression with $n = 20$ training samples. For each plot, the black dots are the training samples, the green curves are the true function, and the blue curves are the prediction function. Each plot corresponds to a different polynomial degree. The degree increases from the left to the right. For the smallest degrees, the training and the test errors are high. The training error is low for the highest degrees, while the test error is high.

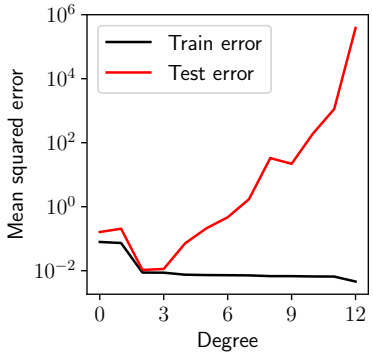


Figure 1.2: The train and test errors for the polynomial regression problem. The train error decreases with the degree of the polynomial. The test error decreases until the degree 3 and then increases, showing the overfitting phenomenon.

another set of parameters $\lambda \in \Lambda$ leading to the following training problem

$$\min_{\theta \in \Theta} g(\lambda, \theta) = \underbrace{\frac{1}{n} \sum_{i=1}^{n} \ell(y_i^{\text{train}}, h_\theta(x_i^{\text{train}}))}_{\text{data fidelity}} + \underbrace{\Omega(\lambda, \theta)}_{\text{regularization}} \quad . \tag{1.8}$$

In Problem (1.8), the training loss comprises two terms. The first term is the data fidelity term, which ensures the prediction fits the training samples well. The second term is the regularization that promotes some property on the learned parameter. Note that different regularization parameter values λ yield different solutions of Problem (1.8). We make this dependency explicit by denoting $\theta^*(\lambda)$ as the minimizer of $g(\lambda, \cdot)$. Several kinds of regularization are proposed in the literature depending on the properties we want to enforce to the learned parameter $\theta^*(\lambda)$. The most simple is the ℓ^2-regularization proposed by Tikhonov and Arsenin (1977) where $\Omega(\lambda, \theta) = \frac{\lambda}{2}\|\theta\|^2$ with $\lambda \in \mathbb{R}^+$. In the context of least squares with ℓ^2-regularization, we talk about *Ridge regression* (Hoerl and Kennard, 1970). In deep learning, the ℓ^2-regularization is often referred to as *weight decay* (Krogh and Hertz, 1991). It reduces the magnitude of the solution $\theta^*(\lambda)$, yielding less variance in the prediction. The ℓ^1-regularization was also proposed by Tibshirani (1996) and is defined by $\Omega(\lambda, \theta) = \lambda\|\theta\|_1$. It has the property to promote sparse solutions, that is, solutions with many zero entries.

1.1.3 Hyperparameter selection as a bilevel optimization problem

For each $\lambda \in \Lambda$, the learned parameter $\theta^*(\lambda)$ is different, leading to different generalization performances of the learned model. This hyperparameter has to be set before training the model. This

choice of the hyperparameter is the hyperparameter selection problem. It is generally chosen to minimize some criterion.

Different criteria are considered in the literature. The hold-out loss we already mentioned in Equation (1.6) (Devroye and Wagner, 1979) is based on the split of the dataset into a training set and a validation set. It reads

$$\Phi(\lambda) = \frac{1}{m}\sum_{j=1}^{m} \ell(y_j^{\text{val}}, h_{\theta^*(\lambda)}(x_j^{\text{val}}))$$
$$\text{where} \quad \theta^*(\lambda) = \arg\min_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} \ell(y_i^{\text{train}}, h_\theta(x_i^{\text{train}})) + \Omega(\lambda,\theta) \ .$$

The cross-validation loss (Stone, 1974; Geisser, 1974) consists in splitting the dataset into K distinct datasets $\mathcal{D}_1, \dots, \mathcal{D}_K$, named folds. For each fold, the model is trained on the remaining folds and evaluated on the current fold. The cross-validation loss is the average of the hold-out losses on the K folds:

$$\Phi(\lambda) = \frac{1}{K}\sum_{k=1}^{K} \frac{1}{|\mathcal{D}_k|} \sum_{(x,y)\in\mathcal{D}_k} \ell(y, h_{\theta_k^*(\lambda)}(x),)$$
$$\text{where} \quad \theta_k^*(\lambda) = \arg\min_{\theta\in\Theta} \frac{1}{\sum_{\substack{s=1\\ s\neq k}}^{K} |\mathcal{D}_s|} \sum_{\substack{s=1\\ s\neq k}}^{K} \sum_{(x,y)\in\mathcal{D}_s} \ell(y, h_\theta(x)) + \Omega(\lambda,\theta) \ .$$

Other criteria include Stein's Unbiased Risk Estimator (SURE) (Stein, 1981), Akaike Information Criterion (AIC) (Akaike, 1974), and Bayesian Information Criterion (BIC) (Schwarz, 1978). SURE, more popular in inverse problem literature (Donoho and Johnstone, 1995; Xiao-Ping Zhang and Desai, 1998; Pesquet et al., 2009), is an unbiased estimator of the model's risk. AIC and BIC are information criteria that consider the model's likelihood and number of parameters, aiming to balance goodness of fit and model complexity.

Let us denote f, the criterion we use to choose the hyperparameter λ. This criterion is always evaluated at $\theta^*(\lambda)$. Thus, the hyperparameter selection problem can be formalized as the following optimization problem

$$\min_{\lambda\in\Lambda} \Phi(\lambda) \triangleq f(\theta^*(\lambda)) \quad \text{s.t.} \quad \theta^*(\lambda) \in \arg\min_{\theta\in\Theta} g(\lambda,\theta) \ . \tag{1.9}$$

In Problem (1.9), one can notice that the definition of the function Φ we want to minimize depends on the solution of an optimization problem which is the problem of learning $\theta^*(\lambda)$ for a given $\lambda \in \Lambda$. This nested optimization problem is a first instance of a bilevel optimization problem. Additional instances of such problems are discussed in Section 1.3.

This formulation as a bilevel problem of the hyperparameter selection was first proposed by Bennett et al. (2006). It has then been studied in the bilevel literature (Pedregosa, 2016; Franceschi et al., 2018; Lorraine et al., 2020), as well as in the inverse problem literature (Calatroni et al., 2016; Pascal et al., 2021; Santambrogio et al., 2024).

1.1.4 From zero-order to first-order methods

The zero-order methods are the most popular approaches to solving Problem (1.9). These methods only use evaluations of the function we want to minimize, that is, Φ in the case of Problem (1.9).

Grid search. The most simple zero-order method is the grid search. It consists in specifying a finite subset of the space of the hyperparameters and making an exhaustive search by evaluating the objective function Φ on all the elements of this subset. This method is implemented in the Python package `scikit-learn` (Pedregosa et al., 2011). An advantage of this method is that the function evaluations

can be parallelized. However, its main drawback is that it scales exponentially with the number of hyperparameters d. Indeed, suppose that our space of hyperparameters is $\Lambda \triangleq [0\,,1]^d$. If we assume that the function Φ is Lipschitz continuous and we want to find a λ that is at a distance less than $\epsilon > 0$ of the optimum λ^*, we have to try $\mathcal{O}((1/\epsilon)^d)$ different values, as illustrated in figure 1.3. Moreover, the grid we set manually is crucial for the performance. If poorly chosen, we can find an "optimal" hyperparameter that is far from the best.

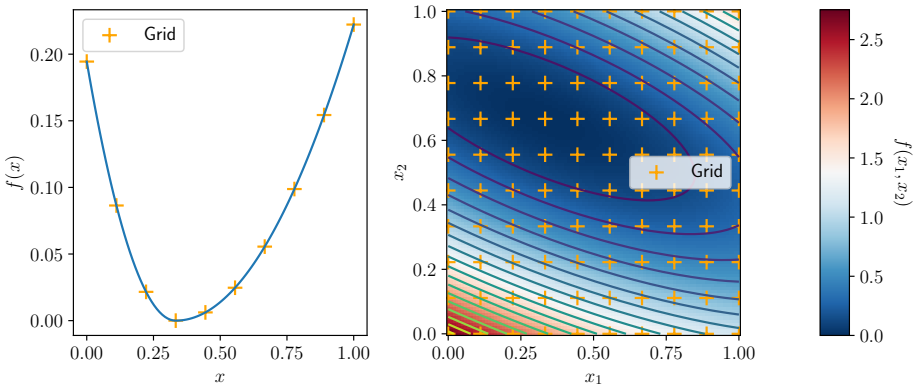


Figure 1.3: Illustation of the curse of dimensionality in zero-order optimization. The orange dots are the test points of the grids. **Left:** Example in dimension 1. For a function defined on $[0, 1]$, one needs 10 test points to find a solution with precision 0.1. **Right:** Example in dimension 2. For a function defined on $[0, 1]^2$, one needs 100 test points to find a solution with the same precision.

Random search. The random search (Bergstra and Bengio, 2012) is a variant of the grid search. Instead of using a deterministic and uniform grid on the hyperparameter space, the test points are chosen randomly according to a probability distribution. On the one hand, it keeps the advantages of simplicity and possible parallelization of the grid search. On the other hand, the randomization enables a smarter exploration of the space if the function Φ is more sensitive in some dimension than in another. This is illustrated in figure 1.4 coming from (Bergstra and Bengio, 2012). Assume that the function Φ is such that $\Phi(\lambda_1, \lambda_2) = \Phi_1(\lambda_1) + \Phi_2(\lambda_2) \approx \Phi_1(\lambda_1)$. Thus, Φ is more sensitive to λ_1 than λ_2. In figure 1.4, Φ_1 is represented in green, and Φ_2 is represented in yellow. With the grid search on the left, only three distinct values of λ_1 are tested while with the random search on the right, nine distinct values of λ_1 are tested, resulting in a better coverage of the subspace.

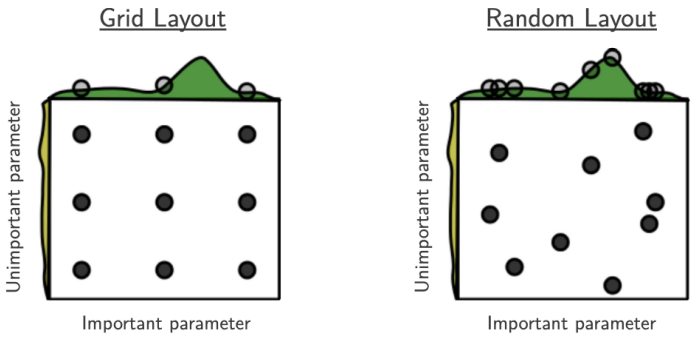


Figure 1.4: Illustrative comparison between grid search and random search borrowed from Bergstra and Bengio (2012). Nine trials of grid search and random search are represented. The function verifies $\Phi(\lambda_1, \lambda_2) = \Phi_1(\lambda_1) + \Phi_2(\lambda_2) \approx \Phi_1(\lambda_1)$. One observes that the grid search explores only three distinct values of λ_1 while the random search explores nine distinct values.

Bayesian optimization. Bayesian optimization methods (Mockus, 1989; Bergstra et al., 2011) refer to a range of zero-order optimization algorithms that take into account the information of the past evaluation of the objective function. It consists in building a probabilistic model of Φ. This is implemented in some packages such as `Optuna` (Akiba et al., 2019) or `Hyperopt` (Bergstra et al., 2015).

All the zero-order methods share a common pitfall: their oracle complexity scales exponentially with the dimension of the outer variable space. A natural question to ask is: is there a zero-order method that has a better oracle complexity? The answer to this question is given by the literature on lower complexity bound in optimization. A lower bound result has always the following form: provided an algorithm class $\mathcal{A}$ and a function class $\mathcal{F}$, there exists a function $f \in \mathcal{F}$ such that any algorithm in $\mathcal{A}$ requires at least $K(\epsilon)$ oracle calls to find a solution with precision ϵ. In the case of zero-order algorithms, the number of oracle calls refers to the number of times the function we want to optimize is evaluated. Nesterov (2018, Theorem 1.1.2) provides a lower bound on the number of function evaluations required to minimize a Lipschitz continuous function over a compact set with zero-order algorithms.

Theorem 1.1. *Consider $\epsilon > 0$ and $L > \frac{\epsilon}{2}$. There exists a function $f : [0,1]^d \to \mathbb{R}$ which verifies*

$$\forall x, y \in [0,1]^d, |f(x) - f(y)| \leq L\|x - y\|_\infty$$

and such that any zero-order algorithm requires at least $\left\lfloor \frac{L}{2\varepsilon} \right\rfloor^d$ function evaluations to find a point x such that

$$f(x) - \inf_{x \in [0,1]^d} f(x) \leq \varepsilon \ .$$

This is all the more problematic in bilevel optimization since evaluating the function Φ requires solving an auxiliary optimization problem which often can not be solved exactly. Thus, in this context, the computation of the function we want to minimize is expensive and inexact.

However, one can remove this dependency of the complexity on the dimension by using higher order information such as gradients or Hessian-vector product (Bubeck, 2015). For this reason, this thesis is devoted to studying and improving first-order methods to solve bilevel optimization problems. As these methods are the core of this thesis, a background chapter is dedicated to them in Chapter 3.

Bilevel optimization problems encompass a broader range of applications beyond hyperparameter tuning. In the next section, we give a more general definition of bilevel optimization problem and provide a brief history of this field.

1.2 Bilevel optimization

1.2.1 General definition and historical perspective

Bilevel optimization problems (Candler and Norton, 1977) are a class of constrained optimization problems where the constraint set involves the solution of another optimization problem. They are formalized as follows:

Definition 1.1 (Bilevel Optimization Problem)**.** A bilevel optimization problem is a constrained optimization problem that takes the following form

$$\begin{aligned} &\min_{x,y \in \mathcal{X} \times \mathcal{Y}} f(x,y) \\ &\text{s.t.} \quad \begin{cases} \forall j \in [J], \psi_j(x,y) \leq 0 \\ y \in \arg\min_{y' \in \mathcal{Y}} \{g(x,y'), \forall i \in [I], \phi_i(x,y') \leq 0\} \end{cases} \end{aligned}$$

where $\mathcal{X} \subset \mathbb{R}^d$, $\mathcal{Y} \subset \mathbb{R}^p$, I and J are positive integers and f, g, $\psi_j's$ and $\phi_i's$ are real-valued functions defined on $\mathcal{X} \times \mathcal{Y}$ that define problem's contraints. The function f is often referred to as the *outer function*, or the *upper-level function*, and the function g is referred to as the *inner function* or the *lower-level function*. The variable x is the *outer variable*, and the variable y is the *inner variable*.

In Definition 1.1, one can observe the hierarchical structure of the problem. For each value of the outer variable $x \in \mathcal{X}$, one has a different inner objective $g(x, \cdot)$, and thus a different constraint set for the inner variable y. Hence, the value of the inner variable y is a response to the value of the outer variable x. In this thesis, we focus on the unconstrained case, where $\mathcal{X} = \mathbb{R}^{d_x}$ and $\mathcal{Y} = \mathbb{R}^{d_y}$ and the

inequality constraints defined by the ψ_j's and the ϕ_i's are absent. This simplifies the problem to the following formulation:

$$\min_{x,y \in \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}} f(x,y) \quad \text{s.t.} \quad y \in \arg\min_{y' \in \mathbb{R}^{d_y}} g(x,y') \ . \tag{1.10}$$

Historically, bilevel problems first emerged in economic science, notably in the context of Stackelberg games (von Stackelberg, 1934, 1952), which models interactions between firms that compete. These games have two players: a leader and a follower. The leader chooses their strategy first. Then, the follower observes the leader's choice and chooses to maximize its payoff. In this context, the outer function f represents the leader's decision function, and the inner function g represents the follower's decision function. Bracken and McGill (1973) were the first to formalize bilevel programs in the operations research literature, although they did not use the term "bilevel.". The term "bilevel" was first introduced by Candler and Norton (1977). Later extensions of bilevel programs included replacing the lower-level optimization problem with a variational inequality constraint (Luo et al., 1996). For a more comprehensive presentation of bilevel optimization in general, we refer readers to the textbooks by Dempe (2011) and Bard (2011).

In the machine learning community, the term "bilevel optimization" first appeared in a paper by Bennett et al. (2006), which proposed using bilevel optimization to select the hyperparameters of support vector machine models. However, earlier works such as Larsen et al. (1996) and Bengio (2000) introduced gradient-based methods for hyperparameter tuning without explicitly referring to bilevel optimization.

Note that the formulation of Problem (1.10) is commonly referred to as *optimistic* bilevel optimization problem (Dempe et al., 2007). In this formulation, the outer function is minimized with respect to the inner variable y. Alternatively, there exists a *pessimistic* formulation of bilevel optimization (Wiesemann et al., 2013; Dempe et al., 2014), where the outer function is maximized with respect to the inner variable y:

$$\min_{x \in \mathbb{R}^{d_x}} \max_{y \in \mathbb{R}^{d_y}} f(x,y) \quad \text{s.t.} \quad y \in \arg\min_{y' \in \mathbb{R}^{d_y}} g(x,y') \ .$$

However, this distinction is beyond the scope of this thesis, as we focus on cases where the inner problem has a unique solution. This specific case will be presented in the next subsection.

1.2.2 Singleton lower-level solution

In some cases, the solution set of the inner problem in Problem (1.10) is a singleton. This is typically the case when the function $g(x, \cdot)$ is strongly convex. In this case, we can define a function $y^* : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ by $y^*(x) = \arg\min_{y \in \mathbb{R}^{d_y}} g(x,y)$ (with a slight abuse of notation by identifying a singleton with its only element).

Definition 1.2 (Value function)**.** Consider Problem (1.10) where $\arg\min_{x \in \mathbb{R}^{d_x}}$ is a singleton for any x. The *value function* Φ is the function defined on $\mathbb{R}^{d_x}$ by

$$\Phi(x) = f(x, y^*(x)) \quad \text{where} \quad y^*(x) = \arg\min_{y \in \mathbb{R}^{d_y}} g(x,y) \ .$$

In that case, Problem (1.10) boils down to

$$\min_{x \in \mathbb{R}^{d_x}} \Phi(x) \ . \tag{1.11}$$

This specific instance of bilevel optimization has received much attention from the machine learning community in recent years. The reason for this success is that under sufficient regularity of the functions f and g, the function Φ is differentiable, enabling the development of gradient-based methods. We will review in more detail these methods in Chapter 3.

Example 1.4 (Ridge regression)**.** *The Ridge regression problem is a regularized version of the least squares problem presented in Example 1.2. It consists in adding an ℓ^2-penalty to the training loss data fitting term. If we keep the notations of Example 1.2, the Ridge regression problem is defined by*

$$\min_{\theta \in \mathbb{R}^p} g(\lambda, \theta) = \underbrace{\frac{1}{2n}\|\mathbf{X}\theta - \mathbf{Y}\|^2}_{\text{data fitting}} + \underbrace{\frac{\lambda}{2}\|\theta\|^2}_{\text{regularization}} \quad . \tag{1.12}$$

for a provided $\lambda > 0$. For any $\lambda > 0$, the function $g(\lambda, \cdot)$ is strongly convex. Thus, the minimizer $\theta^*(\lambda)$ is unique and given by the solution of the normal equation

$$\frac{1}{n}\mathbf{X}^\top \mathbf{X}\theta = \frac{1}{n}\mathbf{X}^\top \mathbf{Y} + \lambda\theta$$

which yields

$$\theta^*(\lambda) = (\mathbf{X}^\top \mathbf{X} + n\lambda \mathbf{I})^{-1}\mathbf{X}^\top \mathbf{Y} \quad .$$

To select the hyperparameter λ, we assume that we have a validation set $\mathcal{D}^{\text{val}} = \{(x_j^{\text{val}}, y_j^{\text{val}})\}_{j=1}^m$ that has not been used during training. We denote by $\mathbf{X}^{\text{val}} = \begin{bmatrix} x_1^{\text{val}} & \dots & x_m^{\text{val}} \end{bmatrix}$ and $\mathbf{Y}^{\text{val}} = (y_1^{\text{val}}, \dots, y_m^{\text{val}})$. The validation loss is given by

$$f(\theta) = \frac{1}{2m}\|\mathbf{X}^{\text{val}}\theta - \mathbf{Y}^{\text{val}}\|^2 \quad . \tag{1.13}$$

The hyperparameter λ is chosen by solving the following bilevel optimization problem

$$\min_{\lambda > 0} \Phi(\lambda) \triangleq f(\theta^*(\lambda)) \quad \textit{s.t.} \quad \theta^*(\lambda) = \operatorname*{arg\,min}_{\theta \in \mathbb{R}^p} g(\lambda, \theta) \quad . \tag{1.14}$$

We plot this function in figure 1.5. The main thing to notice is that the value function Φ is non-convex, despite the inner function g being strongly convex and the outer function f being convex.

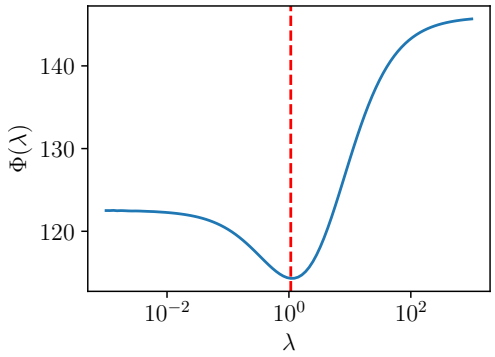


Figure 1.5: Curve of the value function Φ. Here, we generated samples $x \sim \mathcal{N}(0, I_{300})$, a parameter $\theta^* \sim \mathcal{N}(0, I_{300})$ and we generated the target with $y = \langle \theta, x \rangle + 0.1\epsilon$ with $\epsilon \sim \mathcal{N}(0, 1)$. We generated 100 training samples, and 30 validation samples. The dashed red line indicates the value of the hyperparameter λ that minimizes the value function. One observes that the value function is non-convex.

1.2.3 Stochastic problems

In many machine learning applications, such as the hyperparameter selection we have seen in Section 1.1, the functions f and g are defined as expectations:

$$f(x, y) = \mathbb{E}_\xi[f(x, y; \xi)], \quad g(x, y) = \mathbb{E}_\zeta[g(x, y; \zeta)] \tag{1.15}$$

where ξ and ζ are random variables. The empirical risk minimization setting is a particular case of Equation (1.15) where the variables ξ and ζ are indices that are sampled uniformly in a finite set. In this case, the functions f and g in Equation (1.15) take the form of empirical means over samples

$$f(x, y) = \frac{1}{m}\sum_{j=1}^m f_j(x, y), \quad g(x, y) = \frac{1}{n}\sum_{i=1}^n g_i(x, y) \tag{1.16}$$

where m and n are positive integers representing the number of samples in the upper and lower levels. In large sample size cases, evaluating the functions f and g is prohibitive. For this reason, the thesis investigates stochastic algorithms that are more scalable than deterministic ones. Indeed, stochastic algorithms only require a handful of samples to progress in the problem resolution, making them more suitable for large-scale problems. More detailed presentations of these algorithms are provided in subsection 2.2.2 for single-level problems and in Section 3.2 for bilevel problems.

1.3 Other machine learning applications of bilevel optimization

The interest gained by bilevel optimization in the machine learning community is due to its ability to model various machine learning problems. We already presented the hyperparameter selection problem in Section 1.1. In this section, we explore additional machine learning applications that can be effectively addressed using bilevel optimization.

1.3.1 Data reweighting

Consider a training set $\mathcal{D}_{\text{train}} = \{x_i^{\text{train}}\}_{i=1}^n$, where the data samples may not necessarily be labeled, depending on the use case. Sometimes, there can be a discrepancy between the training distribution and the testing conditions. This discrepancy can occur, for instance, when not all training samples are relevant to the downstream task (Grangier et al., 2023), or in a supervised learning context when some labels are corrupted (Franceschi et al., 2017).

A solution to this problem is to assign a weight w_i to each training sample x_i^{train} (Ren et al., 2018; Shu et al., 2019; Wang et al., 2019), leading to the following weighted training loss:

$$g(w, \theta) = \sum_{i=1}^{n} w_i \ell(x_i^{\text{train}}, \theta)$$

where $w \in \mathbb{R}^n$ is a vector of weights, $\theta \in \mathbb{R}^p$ represents the model's parameters, and ℓ is the loss function. These weights serve as data selectors: the highest weights should correspond to important data samples for the downstream task, while the lowest weights correspond to low-quality samples. Alternatively, instead of weighting each sample individually, one can assign a weight per group of samples (or domain) as proposed by Fan et al. (2024b).

To learn those weights, one can parametrize them, for instance by a neural network parametrized by a vector α that takes as input a training sample and returns a weight:

$$w_i = \sigma_\alpha(x_i^{\text{train}}) \ .$$

The problem of learning α can be cast as a bilevel optimization problem. Indeed, assume we have a second dataset $\mathcal{D}_{\text{val}} = \{x_j^{\text{val}}\}_{j=1}^m$, which contains data samples that are specific to the downstream task. As we want the parameter $\theta^*(\alpha)$ to have good performances on the downstream task, we can choose α that solves

$$\begin{aligned} &\min_\alpha f(\theta^*(\alpha)) = \sum_{j=1}^{m} \ell(x_j^{\text{val}}, \theta^*(\alpha)) \\ &\text{s.t.} \quad \theta^*(\alpha) \in \arg\min_\theta g(\alpha, \theta) = \sum_{i=1}^{n} \sigma_\alpha(x_i^{\text{train}}) \ell(x_i^{\text{train}}, \theta) \ . \end{aligned}$$

A classical application of the data reweighting problem is the data hypercleaning task (Franceschi et al., 2017). Assume we have a labeled training set $\mathcal{D}^{\text{train}} = \{(x_i^{\text{train}}, y_i^{\text{train}})\}_{i=1}^n$ and that some training samples may have corrupted labels, as illustrated in figure 1.6 with the MNIST dataset. Conversely, we have a validation set $\mathcal{D}_{\text{val}} = \{(x_i^{\text{val}}, y_i^{\text{val}})\}_{i=1}^n$, where all samples have correct labels. Finding and discarding manually the corrupted samples in the training set can be prohibitive for very large datasets.

The data reweighting framework enables us to avoid this burden by automatically learning which samples are correct and which are corrupted.

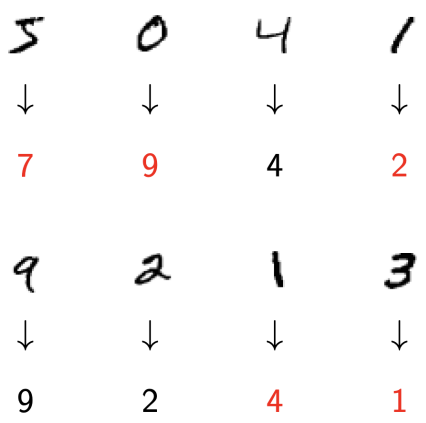


Figure 1.6: Illustration of the labels corruption with the MNIST dataset

1.3.2 Implicit deep learning

Implicit Deep Learning has raised some interest in the last years (Amos and Kolter, 2017; Bai et al., 2019; El Ghaoui et al., 2021) thanks to their memory efficiency by avoiding the memory overhead of the backpropagation. Classical feedforward networks build their prediction with a recurrent procedure

$$z^{k+1} = g_\theta^k(z^k, x)$$

where x is the input, the $(z^k)_{k\in[L]}$ are the intermediate states of the network, and g_θ^k is the k-th layer of the network parametrized by θ (the vector θ includes the parameters of all the layers of the network). To train these networks, one needs to store all the intermediate states $(z^k)_{k\in[L]}$ during the forward pass for the backpropagation step (see subsection 2.3.3). This poses memory issues when training very deep networks.

Deep Equilibrium Models (DEQs) (Bai et al., 2019) consider the case where the same transformation is applied at each layer

$$z^{k+1} = g_\theta(z^k, x) \quad . \tag{1.17}$$

If the function $g_\theta(., x)$ is contractive, the sequence $(z^k)_k$ converges towards a fixed point of the function $g_\theta(., x)$. For this reason, DEQs consist in taking as output of the network the solution $z^*(\theta, x)$ of the equation

$$g_\theta(z^*(\theta, x), x) = z^*(\theta, x) \quad . \tag{1.18}$$

The problem of training a DEQ is a bilevel problem. We want to minimize the training loss under the constraint that we have solved each training sample's fixed-point equation (1.18). This yields the following formulation

$$\begin{aligned} &\min_\theta f(\theta) = \frac{1}{n}\sum_{i=1}^n \ell(y_i^{\text{train}}, z^*(\theta, x_i^{\text{train}})) \\ &\text{s.t.} \quad \forall i \in [n], g_\theta(z^*(\theta, x_i^{\text{train}}), x_i^{\text{train}}) = z^*(\theta, x_i^{\text{train}}) \quad . \end{aligned}$$

The main difference with the problems we have seen so far is that the inner problem is not a minimization problem but a fixed-point problem. Also, this is an instance of a multi-block bilevel problem: we have an inner problem by training sample, and the inner problems are independent of each other. Such formulation enables leveraging implicit differentiation presented in subsection 3.1.2 to train the network. In this way, one avoids the memory burden of the backpropagation.

1.3.3 Neural architecture search

Neural Architecture Search (NAS) (Elsken et al., 2019) aims to find the best neural architecture for a given task. Liu et al. (2019) propose a differentiable way to perform NAS, leading to a bilevel problem.

An architecture (or a cell) is viewed as an acyclic graph where the vertices are represented by an ordered sequence of N nodes $(w_i)_{i\in[N]}$. The vertex w_0 is the input of the cell and w_N its output. Each node is the result of a sum of operations on its parents, that is

$$w_j = \sum_{i<j} o^{(i,j)}(w_i) \ .$$

Therefore, finding the best architecture boils down to finding the best operation set $\{o^{(i,j)}, j \in [N], 1 \leq i < j\}$. We denote $\mathcal{O}$, the (finite) set of operation sets we consider. For $\mathbf{o} \in \mathcal{O}$, let $h_{\mathbf{o},\theta}$ the predictor associated to the operation set $\mathbf{o}$. Finding the best operation set $\mathbf{o}$ can be cast as a bilevel problem with

$$\min_{\mathbf{o}} f(\mathbf{o}, \theta^*(\min_{\mathbf{o}})) = \frac{1}{m}\sum_{j=1}^{m} \ell(y_j^{\text{val}}, h_{\mathbf{o},\theta}(x_j^{\text{val}}))$$
$$\theta^*(\min_{\mathbf{o}}) \in \arg\min_{\theta} g(\mathbf{o}, \theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i^{\text{train}}, h_{\mathbf{o},\theta}(x_i^{\text{train}})) \ .$$

Written in this way, the problem is a combinatorial problem in $\mathbf{o}$. A differentiable reformulation is proposed by Liu et al. (2019), giving rise to Differentiable ARchiTecture Search (DARTS), and it is extended in different fashion (Zhang et al., 2021; Qin et al., 2023; Ye et al., 2023). This is done by considering mixtures of operations where the mixture coefficients are parametrized by a continuous hyperparameter $\boldsymbol{\alpha}$. In DARTS, the operations $\mathbf{o}^{(i,j)}$ are replaced by a weighted sum

$$\bar{o}^{(i,j)} = \sum_{o\in\mathcal{O}^{(i,j)}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o'\in\mathcal{O}^{(i,j)}}\exp(\alpha_{o'}^{(i,j)})} o$$

where $\mathcal{O}^{(i,j)}$ is the set of possible operations between w_j and w_i. Now it is the weights $\boldsymbol{\alpha} = \{\alpha^{(i,j)}\}_{i,j}$ that are optimized, yielding the following bilevel problem

$$\min_{\alpha} f(\boldsymbol{\alpha}, \theta^*(\alpha)) = \frac{1}{m}\sum_{j=1}^{m} \ell(y_j^{\text{val}}, h_{\boldsymbol{\alpha},\theta}(x_i^{\text{val}}))$$
$$\text{s.t.} \quad \theta^*(\alpha) \in \arg\min_{\theta} g(\boldsymbol{\alpha}, \theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i^{\text{train}}, h_{\boldsymbol{\alpha},\theta}(x_i^{\text{train}})) \ .$$

This continuous formulation, illustrated in figure 1.7, opens the door to gradient-based methods such as the ones presented in Chapter 3.

1.4 Outline and contributions of the thesis

The thesis is divided into two parts. In Part I, we provide the necessary background to understand the contributions of the theses. More specifically

- In Chapter 2, we introduce some mathematical tools used in differentiable optimization. We also present some classical first-order optimization algorithms and give some intuition on their behavior. To introduce some proof techniques that will be used later, we recall the proof of some classical convergence results for gradient descent and stochastic gradient descent. We also explain how variance reduction techniques speed up the convergence of stochastic algorithms. Finally, a short introduction to automatic differentiation is given.

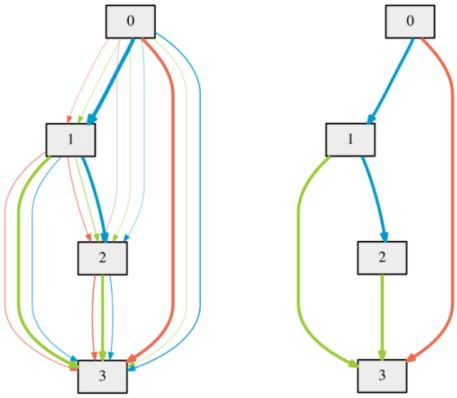


Figure 1.7: Illustration of DARTS. With bilevel optimization, one can optimize jointly mixing coefficients and the network weights. We finally keep the operations with the highest mixing coefficients. This figure is borrowed from (Liu et al., 2019)

- In Chapter 3, we review the main techniques that have been proposed to solve bilevel optimization with gradient-based methods. We first present the implicit differentiation and how it is leveraged in bilevel optimization. In particular, we present some key ingredients that make Approximate Implicit Differentiation (AID) based algorithms work in practice. We then present an alternative approach to implicit differentiation, which consists in differentiating the different steps used to get an approximate solution of the inner problem. This is the iterative differentiation approach (ITD). Finally, we present the penalty-based approaches. These approaches are fully first-order approaches that have gained popularity in the last few years because they do not require second-order information from the inner problem.

In Part II, we present the contributions of the thesis. They are distributed in two chapters:

- In Chapter 4, we tackle bilevel optimization problems where the inner function and the outer function have a structure of empirical means. We introduce a novel framework that consists in maintaining three variables: the inner variable y, the linear system variable v, and the outer variable x. We propose to make them evolve simultaneously following suitable directions that are linear in the outer function and the inner function. This linearity enables easily building stochastic estimators of these directions with a handful of samples. One can easily adapt any stochastic first-order algorithm to bilevel problems thanks to its simplicity and modularity. In this framework, we propose and analyze two instantiations: the Stochastic Bilevel ALgorithm (SOBA), which is an adaptation of stochastic gradient descent to bilevel optimization, and the Stochastic Average Bilevel Algorithm (SABA), which is an adaptation of the SAGA algorithm (Defazio et al., 2014) to bilevel optimization. We show that, under some regularity assumptions, SOBA has a convergence rate in $\mathcal{O}\left(T^{-0.5}\right)$ for constant stepsizes in $\Theta(T^{-0.5})$ (T is the number of iterations) and in $\mathcal{O}\left(\log(T)T^{-0.5}\right)$ with decreasing stepsizes. For SABA, we show that one can converge towards a stationary at the speed of $\mathcal{O}\left((n+m)^{\frac{2}{3}}T^{-1}\right)$ where n and m are the numbers of functions in the outer and inner problems, respectively. We also show that SABA achieves linear convergence under Polyak-Lojasiewicz assumption. Note that these rates match the rate of SGD and SAGA for non-convex single-level problems. This fact motivates Chapter 5.

- In Chapter 5, we interest ourselves in the complexity of bilevel empirical risk minimization. Bilevel empirical risk minimization corresponds to the situation where the inner and the outer functions are empirical means. We propose SRBA, an algorithm in which oracle complexity is $\mathcal{O}\left((n+m)^{\frac{1}{2}}\epsilon^{-1}\right)$ where n and m are the numbers of functions in the outer and inner problems, respectively, and ϵ is the desired precision. We show that this algorithm is near-optimal

in the regime $n \approx m$ among a class of algorithms that includes a wide range of AID-based algorithms. Indeed, we prove that the oracle complexity for this algorithm class is lower bounded by $\Omega\left(m^{\frac{1}{2}}\epsilon^{-1}\right)$ when the inner function is strongly convex.

We finally conclude this thesis in Chapter 6 by summarizing the contributions and discussing some research perspectives.

Part I

Background

CHAPTER 2

BACKGROUND IN FIRST-ORDER OPTIMIZATION

Sommaire

This chapter introduces classical tools in optimization that are useful for this thesis. Most of the results presented here can be found in Nocedal and Wright (2006), Nesterov (2018), or Garrigos and Gower (2023).

2.1 Mathematical background

The design and the analysis of optimization algorithms rely on the properties of the function we want to minimize, often referred to as "the regularity" of the function.

2.1.1 Differentiability and gradient

The differentiability of a function is an important notion that indicates the local behavior of this function. It provides a linear approximation of the function around a point.

Definition 2.1 (Differentiability)**.** Let $f : \mathbb{R}^p \to \mathbb{R}^q$ a function and let $x \in \mathbb{R}^p$. The function f is said to be *differentiable* at x if there exists a linear continuous application $\mathrm{d}f(x) : \mathbb{R}^p \to \mathbb{R}^q$ and a neighborhood $\mathcal{W}_x$ of x such that for any $h \in \mathcal{W}_x$ we have

$$f(x+h) = f(x) + \mathrm{d}f(x)(h) + o(h) \ .$$

If f is differentiable in x, we can consider the matrix of the linear application $\mathrm{d}f(x)$ that we also note $\mathrm{d}f(x)$ with a slight abuse of notation. It belongs to $\mathbb{R}^{q\times p}$ and is called the *Jacobian* of f.

Definition 2.2 (Partial derivative)**.** Let e_i be the ith canonical vector basis of $\mathbb{R}^p$ and suppose that f is real-valued. f has a partial derivative in the direction e_i if $\frac{f(x+te_i)-f(x)}{t}$ has a finite limit when t goes to 0. In this case, we denote

$$\frac{\partial f}{\partial x_i}(x) = \lim_{t\to 0} \frac{f(x+te_i)-f(x)}{t} \quad .$$

When f is differentiable, its Jacobian is given by

$$\mathrm{d}f(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x) & \cdots & \frac{\partial f_i}{\partial x_p}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_q}{\partial x_1}(x) & \cdots & \frac{\partial f_q}{\partial x_p}(x) \end{bmatrix} \in \mathbb{R}^{q\times p} \quad .$$

When f is real-valued, we can introduce the notion of the gradient.

Definition 2.3 (Gradient)**.** Suppose that $q = 1$ and that f is differentiable at x. The *gradient* of f at x is a vector that belongs to $\mathbb{R}^p$ whose components are the partial derivatives of f. It is denoted $\nabla f(x)$, and it is the unique vector that verifies

$$\mathrm{d}f(x).h = \langle \nabla f(x), h\rangle \quad .$$

The components of this vector are the partial derivatives of the function f

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_p}(x) \end{bmatrix} \quad .$$

As illustrated in figure 2.1, the gradient provides the steepest ascent direction of the function at a point x. It is an essential tool in optimization. Indeed, moving in the opposite direction of a function's gradient ensures a decrease in this function.

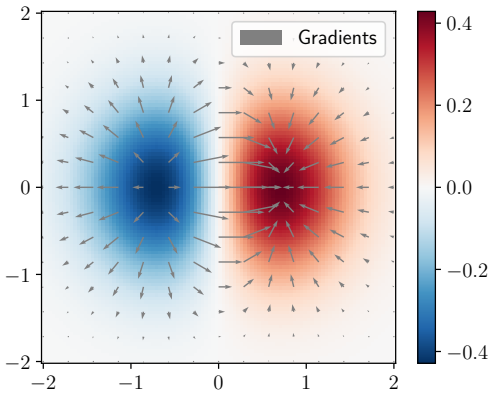


Figure 2.1: Gradient field of the function defined by $f(x,y) = xe^{-(x^2+y^2)}$. Each arrow represents the function's gradient at a point (x,y). The gradient points towards the steepest ascent direction.

When we have a differentiable function, we have a necessary first-order condition for the local minimums of f. This condition is essential for implicit differentiation, so we recall it in Proposition 2.1.

Proposition 2.1. *Let $f : \mathbb{R}^p \to \mathbb{R}$ a differentiable function with a local minimum at x. Then*

$$\nabla f(x) = 0 \quad .$$

Note that the converse is not true. For instance, the function $f(x) = x^3$ has a stationary point at $x = 0$ but is not a local minimum. We will see in the following section that when the function is convex, then the converse of Proposition 2.1 holds.

The Hessian matrix of a function is also an important tool that contains the second-order information of a function.

Definition 2.4 (Hessian)**.** Let $f : \mathbb{R}^p \to \mathbb{R}^q$. We say that f is twice differentiable at $x \in \mathbb{R}^p$ if f and $\mathrm{d}f$ are differentiable. The differential of $\mathrm{d}f$ at x is denoted $\mathrm{d}^2 f(x)$. This is a linear application with values in the space of linear applications between $\mathbb{R}^p$ and $\mathbb{R}^q$. When f is real-valued, we can consider its *Hessian matrix*

$$\nabla^2 f(x) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(x) \right)_{1 \leqslant i,j \leqslant p} \in \mathbb{R}^{p \times p} \ .$$

2.1.2 Convex analysis

Let $f : \mathbb{R}^p \to \mathbb{R}$ a differentiable function. We have seen in Proposition 2.1 that having $\nabla f(x) = 0$ is necessary for x to be a local minimum. For some functions, this condition is also sufficient. This is the case for convex functions, making them very convenient to optimize.

Definition 2.5 (Convexity)**.** Let $f : \mathbb{R}^p \to \mathbb{R}$ a function. We say that f is *convex* if

$$\forall x, y \in \mathbb{R}^p, \forall t \in [0,1], f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) \ .$$

If f is differentiable, the convexity of f is equivalent to

$$\forall x, y \in \mathbb{R}^p, f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle \ .$$

If f is twice differentiable, the convexity of f is equivalent to

$$\forall x \in \mathbb{R}^p, \nabla^2 f(x) \succeq 0 \ .$$

The second characterization of Definition 2.5 provides a geometrical interpretation of the convexity for differentiable functions. A differentiable function is convex if and only if its graph is above its tangents at any point, as illustrated in figure 2.2. Moreover, the condition on the positiveness of the Hessian matrix tells us that the function is convex if and only if it is locally lower bounded by a parabola.

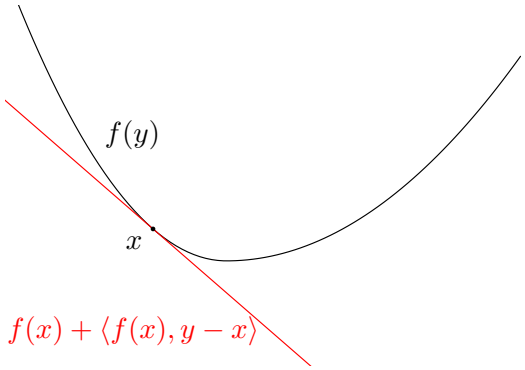


Figure 2.2: Illustation of the convexity of a function. The tangent $y \mapsto f(x) + \langle \nabla f(x), y - x \rangle$ is below the curve of f.

When a function is convex, the first-order condition of Proposition 2.1 is also sufficient for a point to be a global minimum.

Proposition 2.2. *Let $f : \mathbb{R}^p \to \mathbb{R}$ be a convex and differentiable function. Then, $x^* \in \mathbb{R}^p$ is a global minimizer of f if and only if $\nabla f(x^*) = 0$.*

Thanks to this characterization, minimizing a convex function can be recast as the resolution of a system of nonlinear equations. Beyond convexity, strong convexity is a property that makes a function globally lower bounded by a quadratic function.

Definition 2.6 (Strong convexity)**.** Let $f : \mathbb{R}^p \to \mathbb{R}$ be a differentiable function. We say that f is μ-*strongly convex* with parameter $\mu > 0$ if

$$\forall x, y \in \mathbb{R}^p, f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|^2 \ .$$

If f is twice differentiable, the strong convexity of f is equivalent to

$$\forall x \in \mathbb{R}^p, \nabla^2 f(x) \succeq \mu I_p \ .$$

A strongly convex function cannot be flat compared to convex functions. We will see in the following that this property enables faster convergence rates for optimization algorithms. Strongly convex functions are lower bounded by a quadratic function, as illustrated in figure 2.3.

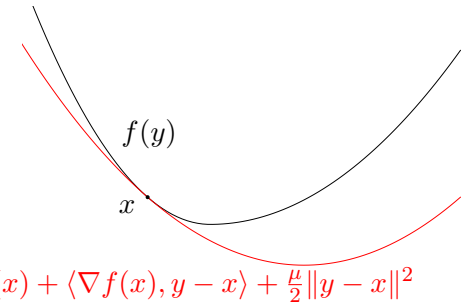


Figure 2.3: Illustation of the strong convexity of a function. The function is lower bounded by the quadratic function $y \mapsto f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|^2$ for a given $x \in \mathbb{R}^p$.

2.1.3 Lipschitz continuity and smoothness

The Lipschitz continuity provides a bound on the sensitivity of a function.

Definition 2.7 (Lipschitz continuity)**.** A function $f : \mathbb{R}^p \to \mathbb{R}^q$ is said to be L-*Lipschitz continuous* if

$$\forall x, y \in \mathbb{R}^p, \|f(x) - f(y)\| \leq L\|x - y\| \ .$$

An essential property of functions is their smoothness. Informally, a small displacement in the input space induces a slight change in the function's gradient.

Definition 2.8 (Smoothness)**.** Let $f : \mathbb{R}^p \to \mathbb{R}$ be a differentiable function. We they that f is L-smooth if its gradient ∇f is L-Lipschitz continuous, that is

$$\forall x, y \in \mathbb{R}^p, \|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \ .$$

In differentiable optimization, a function's smoothness is often a starting point for deriving a descent lemma, which is generally a starting point for analyzing an optimization algorithm.

Proposition 2.3. *Let $f : \mathbb{R}^p \to \mathbb{R}$ be a L-smooth function. The function f is L-smooth if and only if for any $x, y \in \mathbb{R}^p$, it holds*

$$|f(y) - f(x) - \langle \nabla f(x), y - x \rangle| \leq \frac{L}{2}\|y - x\|^2 \ .$$

Geometrically, Proposition 2.3 tells us that a smooth function is upper bounded by a quadratic function, as illustrated in figure 2.4.

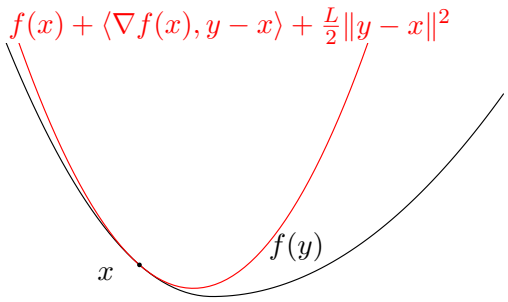


Figure 2.4: Illustation of the smoothness of a function. The function is upper bounded by the quadratic function $y \mapsto f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2$ for a given $x \in \mathbb{R}^p$.

2.2 Gradient descent and variants

2.2.1 Deterministic optimization

Gradient descent (Cauchy, 1847) is the workhorse algorithm to minimize a differentiable function. In this section, let us consider a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ and the following problem

$$\min_{x \in \mathbb{R}^d} f(x) \ . \tag{2.1}$$

If we are at a point $x \in \mathbb{R}^d$ and want to move in some direction δ while decreasing the value of f, the most natural choice for the direction δ is to follow the steepest descent direction. This direction is the opposite direction of the gradient $-\nabla f(x)$. The gradient descent algorithm consists of following this direction at each iteration. It is summarized in Algorithm 1.

Algorithm 1 Gradient descent

Input: initialization $x^0 \in \mathbb{R}^d$ number of iterations T, step size sequence $(\eta^t)_{t<T}$.
for $t = 0, \dots, T-1$ **do**

$$x^{t+1} = x^t - \eta^t \nabla f(x^t) \ .$$

end for

This algorithm seeks a stationary point of the function f while decreasing its value. Indeed, if we choose a constant step size sequence and if the gradient ∇f is continuous, one can observe that in case of convergence of the iterates towards a point x^*, one has necessarily $\nabla f(x^*) = 0$. If we assume the function f to be convex, the point x^* is a global minimizer of the function f according to Proposition 2.2.

A critical property of the gradient descent algorithm is that we have a descent guarantee if we assume that the function f is L-smooth (see Definition 2.8) and for a convenient choice of step sizes. This property is often referred to as a descent lemma in optimization.

Lemma 2.1 (Descent Lemma for gradient descent)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be an L-smooth function and $(x^t)_{t \geq 0}$ a sequence verifying*

$$x^{t+1} = x^t - \eta \nabla f(x^t)$$

with $\eta \in \left(0, \frac{1}{L}\right]$. Then it holds

$$f(x^{t+1}) \leq f(x^t) - \frac{\eta}{2}\|\nabla f(x^t)\|^2 \leq f(x^t) \ .$$

Proof. From Proposition 2.3, we have

$$\begin{aligned} f(x^{t+1}) &\leq f(x^t) + \langle \nabla f(x^t), x^{t+1} - x^t \rangle + \frac{L}{2}\|x^{t+1} - x^t\|^2 \\ &\leq f(x^t) - \eta\|\nabla f(x^t)\|^2 + \frac{L\eta^2}{2}\|\nabla f(x^t)\|^2 \\ &= f(x^t) - \eta\left(1 - \frac{L\eta}{2}\right)\|\nabla f(x^t)\|^2 \ . \end{aligned}$$

Using $\eta \leq 1/L$, we have $-\eta\left(1 - \frac{L\eta}{2}\right) \leq -\frac{\eta}{2}$ yielding the result. □

From this lemma, a non-asymptotic convergence rate for the gradient descent algorithm in the convex case can be derived.

Theorem 2.1 (Convergence rate of gradient descent, convex case)**.** *Assume that the function f is L-smooth, convex and that $\arg\min f$ is non-empty. Let $(x^t)_{t\geq 0}$ be the sequence generated by Algorithm 1 with a constant step size $\eta \in \left(0, \frac{1}{L}\right]$. Then it holds for $x^* \in \arg\min f$ and $f^* = f(x^*)$*

$$f(x^t) - f^* \leq \frac{\|x^0 - x^*\|^2}{2\eta t} \ .$$

The proof of this result uses a classical technique for analyzing a differentiable optimization algorithm. It consists of finding a suitable Lyapunov function that decreases at each iteration. This proof technique will be helpful in Chapter 4 and Chapter 5.

Proof. Let $x^* \in \arg\min f$ and $f^* = f(x^*)$. First we show that the sequence $(\|x^t - x^*\|^2)_{t\geq 0}$ is decreasing. By expanding the square, we have

$$\begin{aligned} \|x^{t+1} - x^*\|^2 &= \|(x^{t+1} - x^t) + (x^t - x^*)\|^2 \\ &= \|x^t - x^*\|^2 + 2\langle x^{t+1} - x^t, x^t - x^* \rangle + \|x^{t+1} - x^t\|^2 \\ &= \|x^t - x^*\|^2 + 2\langle x^{t+1} - x^t, x^{t+1} - x^* \rangle + 2\langle x^{t+1} - x^t, x^t - x^{t+1} \rangle + \|x^{t+1} - x^t\|^2 \\ &= \|x^t - x^*\|^2 + 2\langle x^{t+1} - x^t, x^{t+1} - x^* \rangle - \|x^{t+1} - x^t\|^2 \\ &= \|x^t - x^*\|^2 - 2\eta\langle \nabla f(x^t), x^{t+1} - x^t \rangle - 2\eta\langle \nabla f(x^t), x^t - x^* \rangle - \|x^{t+1} - x^t\|^2 \ . \end{aligned} \tag{2.2}$$

By the smoothness of f, we have

$$-2\eta\langle \nabla f(x^t), x^{t+1} - x^t \rangle \leq -2\eta(f(x^{t+1}) - f(x^t)) + \eta L\|x^{t+1} - x^t\|^2 \ . \tag{2.3}$$

The convexity of f implies

$$-2\eta\langle \nabla f(x^t), x^t - x^* \rangle \leq -2\eta(f(x^t) - f^*) \ . \tag{2.4}$$

Summing Equation (2.3) and Equation (2.4) and injecting in Equation (2.2) yield

$$\begin{aligned} \|x^{t+1} - x^*\|^2 &\leq \|x^t - x^*\|^2 - 2\eta(f(x^{t+1}) - f^*) + (\eta L - 1)\|x^{t+1} - x^t\|^2 \\ &\leq \|x^t - x^*\|^2 - 2\eta(f(x^{t+1}) - f^*) \ . \end{aligned} \tag{2.5}$$

where we used $\eta \leq \frac{1}{L}$.

Let us consider the Lyapunov function $\mathcal{L}^t = t(f(x^t) - f^*) + \frac{\|x^t - x^*\|^2}{2\eta}$. We have

$$\mathcal{L}^{t+1} = (t+1)(f(x^{t+1}) - f^*) + \frac{\|x^{t+1} - x^*\|^2}{2\eta} \ .$$

Using Equation (2.5), we get

$$\begin{aligned}\mathcal{L}^{t+1} &\leq (t+1)(f(x^{t+1}) - f^*) + \frac{\|x^t - x^*\|^2}{2\eta} - (f(x^{t+1}) - f^*) \\ &\leq t(f(x^{t+1}) - f^*) + \mathcal{L}^t - t(f(x^t) - f^*) \\ &\leq t(f(x^{t+1}) - f(x^t)) + \mathcal{L}^t \ . \end{aligned} \tag{2.6}$$

Using the descent lemma Lemma 2.1, we have $f(x^{t+1}) < f(x^t)$ and thus by Equation (2.6), the Lyapunov function $\mathcal{L}$ is decreasing. This implies that

$$t(f(x^t) - f^*) \leq \mathcal{L}^t \leq \mathcal{L}^0 = \frac{\|x^0 - x^*\|^2}{2\eta}$$

which concludes the proof. □

This result shows that the function values $(f(x^t))_{t\geq 0}$ converge sublinearly towards the optimal value f^*. When we assume the strong convexity of the function f, we can improve the previous result by showing the convergence of the iterates at a linear speed thanks to the non-flatness of the function.

Theorem 2.2 (Convergence rate of gradient descent, strongly convex case)**.** *Assume the function f is L-smooth and μ-strongly convex. Let $(x^t)_{t\geq 0}$ be the sequence generated by Algorithm 1 with a constant step size $\eta \in \left(0, \frac{1}{L}\right]$. Then it holds for $x^* = \arg\min f$*

$$\|x^{t+1} - x^*\| \leq (1 - \eta\mu)^t \|x^0 - x^*\| \ . \tag{2.7}$$

Proof. The proof starts by expanding the square

$$\|x^{t+1} - x^t\|^2 \leq \|x^t - x^*\|^2 - 2\eta\langle \nabla f(x^t), x^t - x^*\rangle + \eta^2 \|\nabla f(x^t)\|^2 \ . \tag{2.8}$$

By strong convexity, we have

$$-2\eta\langle \nabla f(x^t), x^t - x^*\rangle \leq -2\eta(f(x^t) - f^*) - \eta\mu\|x^t - x^*\|^2 \ . \tag{2.9}$$

By using Lemma 2.1 and $\eta \leq \frac{1}{L}$, we have

$$\eta^2 \|\nabla f(x^t)\|^2 \leq 2\eta(f(x^t) - f(x^{t+1})) \ . \tag{2.10}$$

Plugging Equation (2.9) and Equation (2.10) in Equation (2.8) yields

$$\begin{aligned}\|x^{t+1} - x^t\|^2 &\leq (1 - \eta\mu)\|x^t - x^*\|^2 - 2\eta(f(x^{t+1}) - f^*) \\ &\leq (1 - \eta\mu)\|x^t - x^*\|^2 \ . \end{aligned}$$

We conclude the proof by unrolling the previous relation. □

2.2.2 Stochastic optimization

In Empirical Risk Minimization such as in Problem (1.4), the functions we want to minimize is the empirical mean over the dataset:

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) \ . \tag{2.11}$$

When the amount of data is large (meaning that n is high) computing the gradient of f becomes too expensive because it requires computing the gradient of each f_i, making gradient descent inefficient. For this reason, stochastic variants of gradient descent have been designed and are widely used in practice.

Stochastic Gradient Descent. The Stochastic Gradient Descent (SGD) algorithm (Robbins and Monro, 1951) is an adaptation of the gradient descent algorithm. Instead of following the direction of the negative gradient of f, SGD consists of following the direction given by an unbiased stochastic estimator of this gradient, built by using only the gradient of f_i for a randomly sampled $i \in [n]$.

Algorithm 2 Stochastic Gradient descent

Input: initialization $x^0 \in \mathbb{R}^d$ number of iterations T, step size sequence $(\eta^t)_{t<T}$.
for $t = 0, \dots, T-1$ **do**
  Draw i_t uniformly at random in $\{1, \dots, n\}$.
  Update x

$$x^{t+1} = x^t - \eta^t \nabla f_{i_t}(x^t) \ .$$

end for

This algorithm has become very popular in machine learning (Bottou, 2010; Bottou et al., 2018) and successful algorithms in deep learning such as AdaGrad (Duchi et al., 2011) or ADAM (Kingma and Ba, 2015) are variants of SGD.

There are many analyses of Stochastic Gradient Descent in the literature. Non-asymptotic analyses can be found in Moulines and Bach (2011) and Ghadimi and Lan (2013). Let us consider the strongly convex case to understand how SGD behaves compared to gradient descent.

Theorem 2.3. *Assume that each function f_i is L-smooth and convex. We assume, furthermore, that the function f is μ-strongly convex and that there exists $\sigma > 0$ such that for any $x \in \mathbb{R}^p$*

$$\mathbb{E}_{i \sim \mathcal{U}([n])}[\|\nabla f_i(x) - \nabla f(x)\|^2] \leq \sigma^2 \ .$$

Then, if we let $x^* = \arg\min_x f$, the iterates of SGD with a constant step size $\eta \in \left(0, \frac{1}{L}\right]$ verify

$$\mathbb{E}[\|x^{t+1} - x^*\|^2] \leq (1 - \eta\mu)^t \|x^0 - x^*\|^2 + \frac{\eta\sigma^2}{\mu} \ . \tag{2.12}$$

Proof. As previously, we start by expanding the square

$$\|x^{t+1} - x^t\|^2 \leq \|x^t - x^*\|^2 - 2\eta\langle \nabla f_{i_t}(x^t), x^t - x^* \rangle + \eta^2 \|\nabla f_{i_t}(x^t)\|^2 \ .$$

We denote $\mathbb{E}_t$ as the conditional expectation given the iterates $x_0, \dots, x_t$. Since $\mathbb{E}_t[\nabla f_{i_t}(x^t)] = \nabla f(x^t)$, we have

$$\mathbb{E}_t[\|x^{t+1} - x^t\|^2] \leq \|x^t - x^*\|^2 - 2\eta\langle \nabla f(x^t), x^t - x^* \rangle + \eta^2 \mathbb{E}_t[\|\nabla f_{i_t}(x^t)\|^2] \ . \tag{2.13}$$

By strong convexity, we have

$$-2\eta\langle \nabla f(x^t), x^t - x^* \rangle \leq -2\eta(f(x^t) - f^*) - \eta\mu\|x^t - x^*\|^2 \ . \tag{2.14}$$

Also,

$$\begin{aligned} \mathbb{E}_t[\|\nabla f_{i_t}(x^t)\|^2] &= \mathbb{E}_t[\|\nabla f_{i_t}(x^t) - \nabla f(x^t)\|^2] + \|\nabla f(x^t)\|^2 \\ &\leq \sigma^2 + \|\nabla f(x^t)\|^2 \ . \end{aligned} \tag{2.15}$$

Plugging Equation (2.14) and Equation (2.15) in Equation (2.13) yields

$$\mathbb{E}_t[\|x^{t+1} - x^t\|^2] \leq (1 - \eta\mu)\|x^t - x^*\|^2 - 2\eta(f(x^t) - f^*) + \eta^2\sigma^2 + \eta^2\|\nabla f(x^t)\|^2 \ .$$

By using Lemma 2.1, since $\eta \leq \frac{1}{L}$, we have

$$\eta^2\|\nabla f(x^t)\|^2 \leq 2\eta(f(x^t - \eta\nabla f(x^t)) - f(x^t)) \ .$$

This yields

$$\begin{aligned}\mathbb{E}_t[\|x^{t+1} - x^t\|^2] &\leq (1-\eta\mu)\|x^t - x^*\|^2 - 2\eta(f(x^t - \eta\nabla f(x^t)) - f(x^*)) + \eta^2\sigma^2 \\ &\leq (1-\eta\mu)\|x^t - x^*\|^2 + \eta^2\sigma^2 \ .\end{aligned}$$

Taking the expectation and unrolling the previous relation yields

$$\mathbb{E}[\|x^{t+1} - x^*\|^2] \leq (1-\eta\mu)^t\mathbb{E}[\|x^0 - x^*\|^2] + \frac{\eta\sigma^2}{\mu} \ .$$

□

When comparing the convergence rates of gradient descent in Equation (2.7) and of SGD in Equation (2.12) for strongly convex functions, one observes the appearance of the variance term $\frac{\eta\sigma^2}{\mu}$. This variance term prevents SGD from converging when run with fixed step sizes. Indeed, in this case, the iterates of SGD oscillate around the optimal point x^* without converging. Therefore, the sequence $(\mathbb{E}[\|x^t - x^*\|^2])_{t\geq 0}$ reaches a plateau. This behavior is illustrated in figure 2.5, where we compare the behavior of gradient descent and SGD on a least squares problem with different step size choices. We observe the linear convergence of gradient descent's iterates. For SGD with fixed step sizes, there are two regimes: a first regime where the distance between the iterates and the solution decreases faster than gradient descent, and a second regime where this distance reaches a plateau. Additionally, the higher the step size, the higher the plateau. One common strategy to avoid reaching this plateau is to use a decreasing step size sequence. Indeed, the analysis of Moulines and Bach (2011) shows convergence in this case. However, this leads to very slow convergence, as we can see with the red curve in figure 2.5.

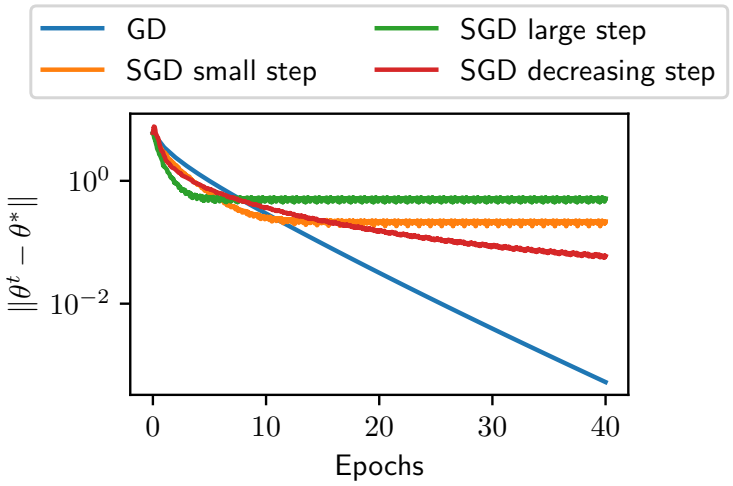


Figure 2.5: Illustration of the behavior of GD and SGD with different step sizes on a least squares problem with 100 samples and 30 features. We plot the distance between the current iterate and the solution in function of the number of epochs (*i.e.*, the number of passes on the dataset). Gradient descent is run with the step size $\frac{1}{L}$. SGD with a large step size uses a constant step size of 0.01. SGD with a small step size uses a constant step size of 0.005. SGD with decreasing step size uses a step size in $\frac{0.07}{\sqrt{t+1}}$ where t is the iteration number. We observe the linear convergence of gradient descent's iterates. SGD with constant step sizes reaches a plateau higher with a larger step size due to the variance of the gradient estimate. With decreasing step size, SGD performs a very slow convergence.

Acceleration with variance reduction. We saw in the previous paragraph that the variance of the gradient estimation prevents SGD from converging. Instead of using vanishing step sizes, it is possible to make this variance go towards zero. This is the idea of the variance reduction.

A typical variance-reduced optimization algorithm example is SAGA (Defazio et al., 2014). SAGA works by maintaining a memory $(g_i^t)_{i\in[n]}$ of the past gradients. More precisely, at iteration t, a random index

i_t is drawn. Then the memory is updated by $g_{i_t}^t = \nabla f_{i_t}(x^t)$ and $g_i^t = g_i^{t-1}$ for $i \neq i_t$. The gradient of the objective function is then estimated by $\nabla f_{i_t}(x^t) - g_{i_t}^{t-1} + \frac{1}{n}\sum_{i=1}^n g_i^{t-1}$. This is an unbiased estimator of the gradient of the objective function because

$$\mathbb{E}_{i_t \sim \mathcal{U}([n])}\left[\nabla f_{i_t}(x^t) - g_{i_t}^{t-1} + \frac{1}{n}\sum_{i=1}^n g_i^{t-1}\right] = \nabla f(x^t) - \frac{1}{n}\sum_{i=1}^n g_i^{t-1} + \frac{1}{n}\sum_{i=1}^n g_i^{t-1} = \nabla f(x^t) \ . \tag{2.16}$$

The SAGA algorithm is summarized in Algorithm 3. The main drawback of SAGA is its memory footprint: it requires storing the n gradients ∇f_i. In practice, the sum $\frac{1}{n}\sum_{i=1}^n g_i^{t-1}$ is also stored so one can update it by rolling average.

Algorithm 3 SAGA

Input: initialization $x^0 \in \mathbb{R}^d$ number of iterations T, step size η.
for $t = 0, \dots, T-1$ **do**
Draw i_t uniformly at random in $\{1, \dots, n\}$.
Set

$$g_i^t = \begin{cases} \nabla f_{i_t}(x^t) & \text{if } i = i_t \\ g_i^{t-1} & \text{otherwise} \end{cases}$$

Update x

$$x^{t+1} = x^t - \eta\left[\nabla f_{i_t}(x^t) - g_{i_t}^{t-1} + \frac{1}{n}\sum_{i=1}^n g_i^{t-1}\right] \ .$$

end for

We provide in Theorem 2.4 a convergence result for SAGA in the strongly convex case coming from Defazio et al. (2014).

Theorem 2.4 (Defazio et al. (2014, Corollary 1))**.** *Assume that each function f_i is L-smooth and μ-strongly convex. With a step size $\eta = \frac{1}{2(\mu n + L)}$, the iterates $(x^t)_{t\geq 0}$ of SAGA verifies*

$$\mathbb{E}[\|x^{t+1} - x^*\|^2] \leq (1 - \eta\mu)^t \left[\|x^0 - x^*\|^2 + \frac{n}{\mu n + L}(f(x^0) - f(x^*))\right] \ .$$

Our observation is that we have a linear convergence of the iterates of SAGA towards the optimal point x^*. Thanks to the variance reduction, the convergence occurs with a fixed step size, making SAGA faster than SGD. A second remark we can make is to scale the step size with respect to the number of samples. Indeed, we have $\eta = \mathcal{O}\left(\frac{1}{n}\right)$. This is illustrated in figure 2.6 where we see that, with constant step size, SAGA converges linearly while SGD reaches a plateau.

In the nonconvex case, the convergence rate of SAGA is established by Reddi et al. (2016).

Theorem 2.5 (Reddi et al. (2016, Theorem 3))**.** *Assume that each function f_i is L-smooth, and let us denote $f^* = \inf f$. Then, if we set $\eta = \frac{1}{3Ln^{\frac{2}{3}}}$, the iterates $(x^t)_{t\geq 0}$ of SAGA verify*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[\|\nabla f(x^t)\|^2] \leq \frac{12Ln^{\frac{2}{3}}(f(x^0) - f^*)}{T} \ .$$

In comparison, gradient descent for nonconvex functions achieves $\mathcal{O}\left(\frac{1}{T}\right)$ convergence rate. As we compute n gradient per iteration with gradient descent, this leads to a $\mathcal{O}(n\epsilon^{-1})$ complexity for gradient descent, which is worse than the $\mathcal{O}(n^{\frac{2}{3}}\epsilon^{-1})$ complexity of SAGA in this setting.

Note that a wide variety of variance reduction algorithms exist. The SAG algorithm (Schmidt et al., 2017) is similaire to SAGA but the estimate direction is $\frac{1}{n}\left(\nabla f_{i_t}(x^t) - g_{i_t}^{t-1} + \sum_{i=1}^n g_i^{t-1}\right)$ instead of $\nabla f_{i_t}(x^t) - g_{i_t}^{t-1} + \frac{1}{n}\sum_{i=1}^n g_i^{t-1}$. The SVRG algorithm (Johnson and Zhang, 2013) is a variant of SAGA in

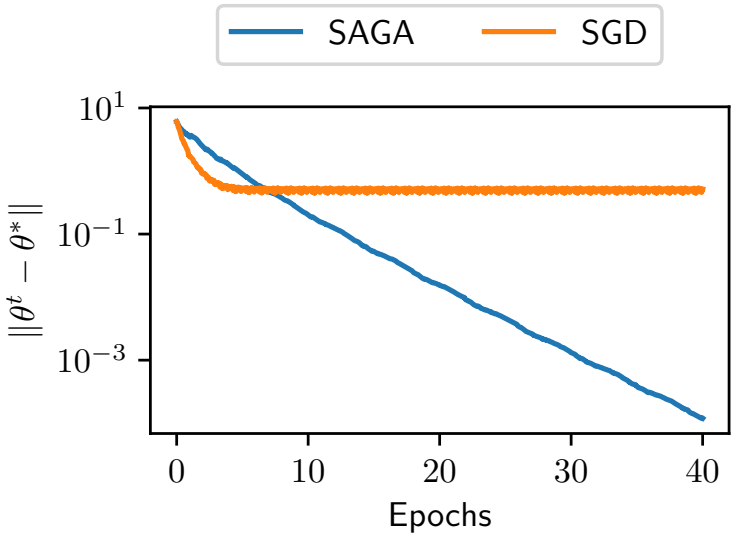


Figure 2.6: Illustration of the behavior of SAGA and SGD on a least squares problem with 100 samples and 30 features. We plot the distance between the current iterate and the solution in function of the number of epochs (*i.e.*, the number of passes on the dataset). SAGA is run with the step size 0.006. SGD uses a constant step size of 0.01. Despite a constant step size, SAGA can converge thanks to the variance reduction. SGD, for its part, reaches a plateau.

which the memories $(g_{i\in[n]}^{t-1}$ are replaced by gradients computed at the same reference point $\tilde{x}$ which is updated periodically. The gradient is then estimated by $\nabla f_{i_t}(x^t) - \nabla f_{i_t}(\tilde{x}) + \frac{1}{n}\sum_{i=1}^n \nabla f_i(\tilde{x})$. By doing so, we do not need anymore to store the n gradients but only the reference point $\tilde{x}$ and the average of the gradients $\frac{1}{n}\sum_{i=1}^n \nabla f_i(\tilde{x})$. However, the number of steps between two updates of $\tilde{x}$ is a hyperparameter that dramatically influences the algorithm's performance.

2.2.3 Complexity measure of optimization algorithms

The complexity of an optimization algorithm is a quantification of the number of elementary operations required to reach a solution with a certain precision. The quality of a solution x returned by an optimization algorithm can be measured in different ways, depending on the problem considered:

- **Distance to the solution.** If the function has a unique minimizer x^*, the (squared) distance to the solution is the quantity $\|x - x^*\|^2$ can be used as a performance measure, as it is the case in Equation (2.7) and Equation (2.12). This performance measure is generally used for strongly convex functions.

- **Suboptimality.** If the function has multiple minimizers, measuring the distance between a candidate solution x and one of the minimizers of the function does not make sense. For this reason, we prefer to use the suboptimality measure $f(x) - \inf f$, as done in Theorem 2.1. This performance measure is generally used for convex functions.

- **Stationarity.** When the function is non-convex, the access to $\inf f$ is not possible in general. For this reason, we generally use the (squared) norm of the function's gradient $\|\nabla f(x)\|^2$. However, having $\|\nabla f(x)\|^2 = 0$ does not mean that x is a global minimizer of f, but this is a necessary condition.

Now that we know how to measure a solution's quality, there are two ways to express the complexity of an optimization algorithm. For a candidate solution x, consider $\mathcal{C}(x)$ one of the above performance measures.

- **Convergence rate.** For an iterative algorithm, a convergence rate is an upper bound on the performance measure $\mathcal{C}(x^t)$ that depends on the iteration number t. More formally, it is a function $\sigma : \mathbb{N} \to \mathbb{R}^+$, such that $\lim_{t\to\infty} \sigma(t) = 0$ and $\mathcal{C}(x^t) \leq \sigma(t)$. It ensures that $\mathcal{C}(x^t)$ converges to zero and quantifies the convergence speed.
- **Oracle complexity.** The complexity of an optimization algorithm is the number of elementary operations required to reach a solution x such that $\mathcal{C}(x) \leq \epsilon$. In differentiable optimization, a proxy of the number of elementary operations is the number of calls to oracles, that is, the number of evaluations of the function and its derivative.

It is generally possible to go from one expression of the complexity to another. For instance, for minimizing a convex finite sum such as Equation (2.11), we showed in Theorem 2.1 that for the iterates of gradient descent verify

$$f(x^t) - \inf f \leq \frac{\|x^0 - x^*\|^2}{2\eta t} \quad .$$

This is a convergence rate result with $\mathcal{C}(x) = f(x) - \inf f$ and $\sigma(t) = \frac{\|x^0 - x^*\|^2}{2\eta t}$. Since at each iteration, we compute n gradients, the total number K of calls to oracles sufficient to have $\mathcal{C}(x^t) \leq \epsilon$ is n multiplied by the smallest t such that $\sigma(t) \leq \epsilon$. This gives $K = \left\lceil \frac{n\|x^0 - x^*\|^2}{2\eta\epsilon} \right\rceil = \mathcal{O}(n\epsilon^{-1})$.

2.3 Automatic differentiation

This section is based on the following blog post we published in the dedicated track at ICLR 2024:

M. Dagréou, P. Ablin, S. Vaiter, T. Moreau. How to compute Hessian-vector products?. In ICLR blogpost track, 2024.[1]

The practical efficiency of optimization algorithms that use derivatives highly relies on the ability to compute these derivatives efficiently and accurately. Automatic differentiation (AD) is a tool that enables this. In what follows, we briefly and informally present its two main modes: forward and reverse. Then, we also describe how we can leverage AD to compute Hessian-vector products (HVP). Finally, we present a benchmark of AD methods to compute HVPs with deep learning architectures.

For a more in-depth introduction, the reader can refer to the book of Griewank and Walther (2008) or the survey of Baydin et al. (2018).

2.3.1 Computational graph

Automatic differentiation relies on the notion of a computational graph (Bauer, 1974). It is a directed acyclic graph that represents the succession of elementary operations required to evaluate a function. A simple computational graph of a function $f : \mathbb{R}^d \to \mathbb{R}^p$ is represented in figure 2.7.

Figure 2.7: Example of computational graph

In this graph, the vertices $z_i \in \mathbb{R}^{m_i}$ represent the intermediate states of the evaluation of f. To get the vertex z_i, we use the values of its parents in the graph z_{i-1}, with simple transfer functions $z_i(z_{i-1})$. The computational complexity of the function evaluation depends on the complexity of the considered graph, as one node might have more than one parent. The memory footprint of the evaluation of the function is also linked to the maximum number of parents that can have a vertex in the computational graph, as their value needs to be stored until all children nodes have been computed.

[1] https://iclr-blogposts.github.io/2024/blog/bench-hvp/

Let us take an example with a multilayer linear perceptron (MLP) with two layers. The function $f_x : \mathbb{R}^h \times \mathbb{R}^{h\times p} \to \mathbb{R}$ is defined for an input $x \in \mathbb{R}^p$ by

$$f_x(U, W) = \frac{1}{2}(UWx)^2 \ . \tag{2.17}$$

Here, the input θ corresponds to the parameters of the network (U, V) and the intermediate steps are $z_1 = Wx$, $z_2 = Uz_1$ and $z_3 = \frac{1}{2}z_2^2$. A possible computational graph to get $f_x(U, W)$ is in the figure 2.8. The associated Python code to compute f_x is

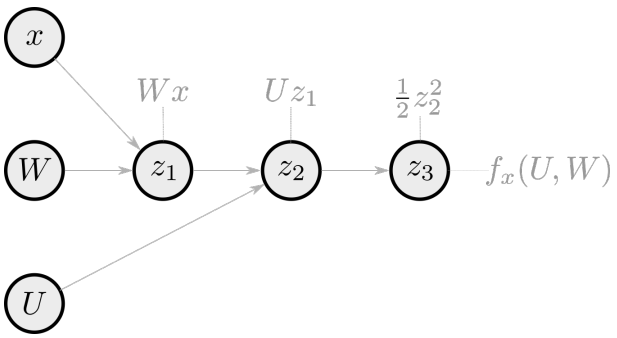


Figure 2.8: Example of computational graph for the MLP

```
def f(U, W):
    z1 = W @ x
    z2 = U @ z1
    z3 = 0.5 * z2**2
    return z3
```

Here, the feed-forward structure of the function makes the computational graph very simple, as each node has a single intermediate result parent.

AD uses this computational graph to compute the function's derivatives. Using the chain rule, the Jacobian $\frac{\partial f}{\partial \theta}(\theta)$ of f is obtained as a product of the Jacobian of the intermediate states $z_1, \ldots, z_n$.

$$\underbrace{\frac{\partial f}{\partial \theta}(\theta)}_{p\times d} = \frac{\partial z_n}{\partial \theta} = \frac{\partial z_n}{\partial z_1}\frac{\partial z_1}{\partial \theta} = \cdots = \underbrace{\frac{\partial z_n}{\partial z_{n-1}}}_{p\times m_{n-1}} \underbrace{\frac{\partial z_{n-1}}{\partial z_{n-2}}}_{m_{n-1}\times m_{n-2}} \cdots \underbrace{\frac{\partial z_1}{\partial \theta}}_{m_1\times d} \ . \tag{2.18}$$

Depending on the order of the multiplication, one can compute the derivative of f with respect to θ in two ways: the forward mode and the reverse mode.

2.3.2 Forward mode

The forward mode of AD was first proposed by Wengert (1964). It allows to compute Jacobian-vector product, that is, the product of the Jacobian of a function $f : \mathbb{R}^d \times p$ with a vector v

$$\frac{\partial f}{\partial \theta}(\theta) \times v = \frac{\partial z_n}{\partial z_{n-1}}\frac{\partial z_{n-1}}{\partial z_{n-2}} \cdots \frac{\partial z_1}{\partial \theta} v \ . \tag{2.19}$$

It consists of doing the multiplications in Equation (2.19) from the right to the left. It is a forward pass in the computational graph where we propagate at the same time the states z_i and the partial derivatives $\frac{\partial z_{i+1}}{\partial z_i}$. If f is real-valued, the ith coordinate of its gradient is exactly given by the product of the Jacobian of f and the ith canonical basis vector e_i since

$$\frac{\partial f}{\partial \theta_i}(\theta) = \lim_{t\to 0} \frac{f(\theta + te_i) - f(\theta)}{t} \ . \tag{2.20}$$

Thus, we can get its gradient by computing each of the d JVPs $\left(\frac{\partial f}{\partial \theta_i}(\theta) \times e_i\right)_{1 \leq i \leq d}$ with forward AD.

To understand properly what is happening when using forward differentiation, let us go back to the linear MLP defined in Equation (2.17). If we implement by ourselves the forward differentiation to get the JVP, we obtain the following code.

```
def jvp(U, W, v_u, v_w):
    # Forward diff of f
    z1 = W @ x
    v_z1 = v_w @ x  # Directional derivative of W -> W @ x in the
   direction v_w

    z2 = U @ z1
    v_z2 = U @ v_z1 + v_u @ z1  #  Directional derivative of (U, z_1) ->
    z2 in the direction (v_u, v_z1)

    v_z3 = v_z2 @ z2  # Directional derivative of z2 -> .5*z2**2 in the
   direction v_z2
    return v_z3
```

In comparison with the code of the evaluation of f_x, there are two more operations corresponding to the computation of the dual variables `v_z1` and `v_z2`. In terms of memory, if we consider the computation of the JVP as coded in the previous snippet, the maximum number of parents of a vertex is four. This maximum is achieved by the vertex `v_z2` which has the vertices `U`, `v_z1`, `v_u` and `z1` as parents.

In `JAX`, we get the JVP of a function f in the direction v with `jax.jvp(f, (params, ), (v, ))[1]`.

2.3.3 Reverse mode

The reverse mode of automatic differentiation (Linnainmaa, 1970, 1976) is the most used in practice in machine learning. Indeed, it is the more efficient mode to compute gradients. In a deep learning context, where we want to compute gradients of the cost function with respect to the model's parameters, the reverse mode is often referred to as backpropagation (Rumelhart et al., 1986).

For $u \in \mathbb{R}^p$, the reverse mode aims at computing VJPs

$$u^\top \frac{\partial f}{\partial \theta}(\theta) = u^\top \frac{\partial z_n}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial z_{n-2}} \cdots \frac{\partial z_1}{\partial \theta} \quad . \tag{2.21}$$

In the reverse AD, the multiplications of Equation (2.19) are done from the left to the right. It requires doing one forward pass in the computational graph to compute the intermediate states z_i and then a backward pass to propagate the successive partial derivatives from the left to the right. Contrary to the forward mode, it has a more important memory footprint. Indeed, it requires storing the values of all the states. For instance, to compute the last term $\frac{\partial z_3}{\partial z_2}$, one needs the value of z_2 which was the first computed during the forward pass. If f is real-valued, u is a scalar, and the VJP is the multiplication of the gradient of f by u. Thus, one can get the gradient on f by using $u = 1$ and performing only one reverse differentiation. This makes this mode more efficient in computing gradients.

Let us observe what happens if we manually code the backpropagation to get the gradient of the previous function f_x defined by $f_x(U, W) = \frac{1}{2}(UWx)^2$.

```
def gradient(U, W):
    # Forward pass
    z1 = W @ x
    z2 = U @ z1
    z3 = 0.5 * z2**2
```

```
    # Reverse pass
    ## Transfer function: z3 = 0.5 * z2**2
    dz2 = z2  # derivative of z3 wrt z2

    ## Transfer function: z2 = U @ z1
    dU = jnp.outer(dz2, z1)  # derivative of z3 wrt U
    dz1 = U.T @ dz2  # derivative of z3 wrt z1

    ## Transfer function: z1 = W @ x
    dW = jnp.outer(dz1, x)   # derivative of z3 wrt W

    return dU, dW
```

This function returns the gradient of f_x. When reading this code, we understand that one needs to store all the intermediate values of the forward pass in the graph. Indeed, if we look at the case of `z1`, which is the first node computed, it is used four steps later for the computation of `dU`.

To get the gradient in JAX, one can use `jax.grad(f)(params)`.

2.3.4 Hessian-vector products

Many algorithms in bilevel optimization require the computation of Hessian-vector products (HVP), that is quantities of the form

$$\nabla^2 f(x)v$$

for a twice differentiable function $f : \mathbb{R}^p \to \mathbb{R}$ and a vector v. At first sight, computing such a quantity seems prohibitive. Indeed, the naive way to compute it is to compute the full Hessian matrix $\nabla^2 f(x)$ and then multiply it by v. This is not doable in high dimensions because storing the Hessian matrix requires $\mathcal{O}(p^2)$ memory.

Automatic differentiation frameworks such as `JAX` (Bradbury et al., 2018) or `PyTorch` (Paszke et al., 2019) enable to compute HVP efficiently without computing the full Hessian matrix. The idea comes from Pearlmutter (1994), who proposes to leverage the following observation: the HVP is also the directional derivative of the gradient in the direction v:

$$\nabla^2 f(x)v = \lim_{t \to 0} \frac{1}{t}[\nabla f(x + tv) - \nabla f(x)] = \nabla[\langle \nabla f(\cdot), v \rangle](x)$$

Based on this identity, AD enables to compute HVPs in three ways, as described in the JAX documentation [2].

Forward-over-reverse. The forward-over-reverse mode consists of doing forward differentiation in a computational graph of the gradient of f. Its implementation in `JAX` is only two lines of code.

```
def hvp_forward_over_reverse(f, params, v):
  return jax.jvp(jax.grad(f), (params, ), (v, ))[1]
```

In this case, `jax.grad(f)(params)` is computed by backward AD, whose complexity is two times the complexity of evaluating f. Thus, the temporal complexity of `hvp_forward_over_reverse` is roughly four times the complexity of the evaluation of f.

To better see what happens, let us consider again our function f_x defined by Equation (2.17). The Python code of the forward-over-reverse HVP follows:

```
def forward_over_reverse(U, W, v_U, v_W):
    # Forward through the forward pass through f
    z1 = W @ x
    v_z1 = v_W @ x
```

[2] https://jax.readthedocs.io/en/latest/notebooks/autodiff_cookbook.html

```
    z2 = U @ z1
    v_z2 = U @ v_z1 + v_U @ z1

    # z3 = 0.5 * z2**2
    # Forward through the backward pass through f
    z4 = z2  # dz2
    v_z4 = v_z2  # v_dz2

    z5 = jnp.outer(z4, z1)  # dU
    v_z5 = jnp.outer(v_z4, z1) + jnp.outer(z4, v_z1)  # v_dU

    z6 = U.T @ z4  # dz1
    v_z6 = U.T @ v_z4 + v_U.T @ z4  # v_dz1

    z7 = jnp.outer(z6, x)  # dW
    v_z7 = jnp.outer(v_z6, x)  # v_dW

    return v_z5, v_z7  # v_dU, v_dW
```

The take-home message of this part is that, after computing the gradient of f_x, one can consider a computational graph of this gradient and perform forward differentiation through this new computational graph. Here, the variables `z1`,..., `z7` are the vertices of a computational graph of the gradient of f_x. The nice thing is that this mode enables the gradient and the HVP to be obtained at the same time. Indeed, in the previous snippet, `z5` and `z7` are the components of the gradient of f_x, which could be also returned if needed. This feature can be useful in bilevel optimization, for instance.

Reverse-over-reverse. Instead of doing forward differentiation of the gradient, one can multiply the gradient by v and thus get a scalar. We can then backpropagate into this scalar product. This is the reverse-over-reverse mode.

It can be implemented by these lines of code.

```
def hvp_reverse_over_reverse(f, params, v):
  return jax.grad(lambda y: jnp.vdot(jax.grad(f)(y), v))(params)
```

Since the gradients are computed by backpropagation, the complexity of `hvp_reverse_over_reverse` is twice the complexity of `jax.grad(f)`, which is roughly four times the complexity of the evaluation of f.

Writting down the code of the reverse-over-reverse HVP for our function f_x defined by Equation (2.17) makes us understand the differences between this mode and the forward-over-reverse mode. Particularly, one can notice that there are more elementary operations in `reverse_over_reverse` mode than in `forward_over_reverse`. In terms of memory footprint, `reverse_over_reverse` requires storing the values of the vertices of the computational graph of the gradient of f_x, while `forward_over_reverse` only needs to store the values of the vertices of the computational graph of f_x. Thus, the former is less efficient than the latter.

```
def reverse_over_reverse(U, W, v_u, v_w):
    # Forward through <grad(f), v>
    ## Forward through f
    z1 = W @ x
    z2 = U @ z1
    z3 = 0.5 * jnp.linalg.norm(z2)**2

    ## Reverse through f
    z4 = z2  # dz2
    z4 = jnp.outer(z3, z1) # dU
```

```
    z5 = U.T @ z3 # dz1
    z6 = jnp.outer(z5, x) # dW

    # Output: dot product <grad(f), v>
    z7 = jnp.sum(z4 * v_u) + jnp.sum(z6 * v_w)

    # Backward through z7 = <grad(f),v>
    ## z7 = jnp.sum(z4 * v_u) + jnp.sum(z6 * v_w)
    dz6 = v_w
    dz4 = v_u

    ## z6 = jnp.outer(z5, x)
    dz5 = dz6 @ x

    ## z5 = U.T @ z3
    dz3 = U @ dz5
    ddU = jnp.outer(z3, dz5)  # Derivative of z7 wrt U

    ## z4 = jnp.outer(z3, z1)
    dz3 += dz4 @ z1
    dz1 = dz4.T @ z3

    ## z3 = z2
    dz2 = dz3

    ## z2 = U @ z1
    dz1 += dz2 * U
    # As U appears multiple times in the graph, we sum its contributions
    ddU += jnp.outer(dz2, z1)

    ## z1 = W @ x
    ddW = jnp.outer(dz1, x)  # Derivative of z7 wrt W

    return ddU, ddW
```

Reverse-over-forward. What about doing forward differentiation of f rather than reverse propagation? This is what is done in the reverse-over-forward mode. It consists in backpropagating in the computational graph of the JVP of f and v.

```
def hvp_reverse_over_forward(f, params, v):
  jvp_fun = lambda params: jax.jvp(f, (params,), (v,))[1]
  return jax.grad(jvp_fun)(params)
```

Since we backpropagate only once, the memory burden is lower than for `hvp_reverse_over_reverse`. In comparison with `hvp_forward_over_reverse`, the complexity is the same. However, one can notice that `hvp_forward_over_reverse` enables computing at the same time the gradient of f and the HVP, which is not the case for the `hvp_reverse_over_reverse` mode.

The code of the reverse-over-forward HVP for the MLP f_x defined by Equation (2.17) is the following.

```
def reverse_over_forward(U, W, v_U, v_W):
    # Forward diff of f to  <grad(f), v>
    z1 = W @ x
    z6 = v_W @ x  # v_z1

    z2 = U @ z1
    z5 = U @ z6 + v_U @ z1  # v_z2

    # output <grad(f), v>
```

```
    z4 = z5 @ z2  # v_z3

    # Backward pass through <grad(f), v>
    ## z4 = z5 @ z2
    dz2 = z5
    dz5 = z2  # dv_z2

    ## z5 = U @ z6 + v_U @ z1
    dz1 = v_U.T @ dz5
    dz6 = U.T @ dz5  # dv_z1
    ddU = jnp.outer(dz5, z6)  # derivative of z4 wrt U

    ## z2 = U @ z1
    # As U and dz1 appear multiple times, we sum their contributions
    dz1 += U.T @ dz2
    ddU += jnp.outer(dz2, z1)

    ## z1 = W @ x
    ddW = jnp.outer(dz1, x)
    return ddU, ddW
```

2.3.5 Benchmarking HVP computation with deep learning architectures.

While these three methods compute the same outputs, the different ways of traversing the computational graph change their overall time and memory complexities. We now compare the computation of HVPs with these three methods for various deep-learning architectures[3]. To cover a broad range of use cases, we consider a residual network ResNet34 (He et al., 2015) and a transformer-based architecture ViT-base (Dosovitskiy et al., 2021) for image classification as well as a transformer for natural language processing Bert-base (Devlin et al., 2019). We use the `Flax` and `PyTorch` implementations of these architectures available in the `transformers` package provided by Hugging Face.

All computations were run on an Nvidia A100 GPU with 40 GB of memory.

Time complexity. The first comparison we make is a comparison in terms of wall-clock time between the different ways to compute HVPs and also the computation of a gradient by backpropagation. For each architecture, we compute the gradient of the model with respect to the parameters by backpropagation. We also compute the HVPs in forward-over-reverse, reverse-over-forward and reverse-over-reverse modes. For each computation, we measure the time taken. Specifically for the HVPs, we subtract the time taken by a gradient computation to get only the time of the overhead required by the HVP computation. The inputs for each architecture are generated randomly. For the ResNet34 architecture, we generated a batch of images of size 224x224x3. To limit out-of-memory issues in the experiments, we generated, for the ViT architecture, images of size 96x96x3. For the BERT architecture, we generated a batch of sequences of length 32.

We first use `JAX` with just-in-time compilation. Each computation is run 90 times. The results are in figure 2.9. We plot, on the left of the figure, the median computation time and also the 20% and 80% percentile in black. The computations are done with a batch size of 128. We observe that the overhead over the gradient computation for the HVP computation is between one and twice the time of a gradient computation for the three architectures. Consequently, a whole HVP computation takes between twice and three times the time of a gradient calculation. This is consistent with the theory. One can notice that the reverse-over-reverse is slightly slower than the others in all the cases. The forward-over-reverse and reverse-over-forward are, as for them, very close in terms of time.

We also report on the right figure the computational time of each method with respect to the batch size

[3]The code of the benchmark is available on https://github.com/MatDag/bench_hvp/

for the ResNet34 architecture. We observe, as expected, that the computational time scales linearly with the batch size.

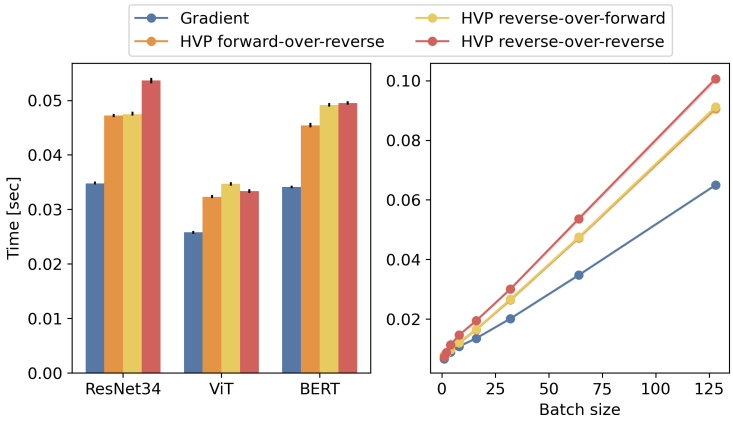


Figure 2.9: **Left:** Median computation time of the gradient and HVPs in `JAX` for the ResNet34, ViT, and BERT architectures with a batch size of 128. The 20% and 80% percentiles are also reported. **Right:** Evolution of the computational time of the gradient and HVPs with the batch size for the ResNet34 architecture.

We run a similar experiment with the functional API available in `PyTorch` `torch.func` similar to the one `JAX` has. The results we show in figure 2.10 are more contrasted.

In the case of ResNet34, the scaling between the different methods is similar to the one we get with `JAX`. Also, during our experiments, we figured out that batch normalization made the forward computation slow and induced out-of-memory issues. Thus, we removed the batch normalization layers from the ResNet34 architecture.

For ViT and BERT, the forward-over-reverse is surprisingly longer than the reverse-over-reverse method. Moreover, the scaling between the gradient and HVP computational time differs from the one we get with `JAX`. Indeed, for these architectures, the HVP computations take between four and five more time than the gradient computations. This is a discrepancy with what we would expect in theory. This might be because, at the time we are writing this blog post, the functional API of `PyTorch` is still in its early stages. Particularly, we could not use the compilation with `torch.compile` because it does not work with some operators of `torch.func` such as `torch.func.jvp`.

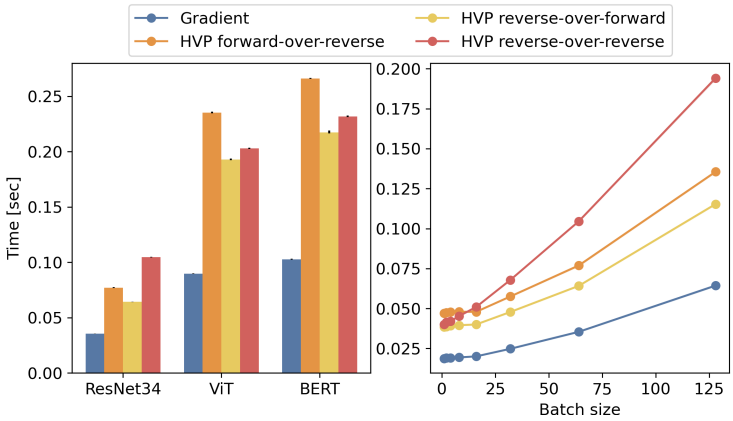


Figure 2.10: **Left:** Median computation time of the gradient and HVPs in `PyTorch` for the ResNet34, ViT, and BERT architectures with a batch size of 128. The 20% and 80% percentiles are also reported. **Right:** Evolution of the computational time of the gradient and HVPs with the batch size for the ResNet34 architecture.

Memory complexity. We also compare the memory footprint of each approach. The figure 2.11 provides the results we get with `JAX` jitted code. On the left, we represent the result for each method and model with a batch size of 64. On the right, we show the evolution of the memory footprint of each method for the ResNet34 with the batch size. Surprisingly, we could observe that the memory footprint of the different methods to compute HVPs does not vary for a given model. This is counterintuitive since we expect that the reverse-over-reverse method has a larger memory footprint due to the double backpropagation.

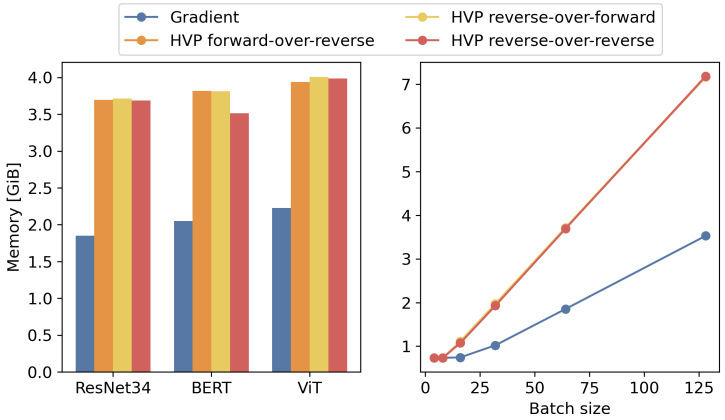


Figure 2.11: **Left:** Memory footprint of the gradient and JVP computation in `JAX` jitted code. **Right:** Evolution of the memory footprint with de batch size

However, we do the same experiment by *disabling the JIT compilation*. The result we get corroborates the theory. One can observe in the figure 2.13 that the memory footprint of the reverse-over-reverse method is larger than the one of the forward-over-reverse and reverse-over-forward methods. This is because the reverse-over-reverse involves two successive backward differentiations while the other two involve only one reverse differentiation. Moreover, it scales linearly with the batch size, which was not the case in the previous figure in the small batch size regime.

In light of these two results, the clever memory allocation performed during just-in-time compilation significantly reduces the memory footprint of the HVP computations.

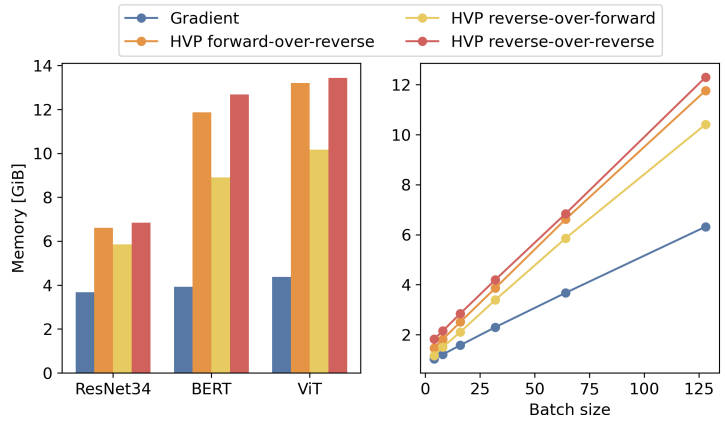


Figure 2.12: **Left:** Memory footprint of the gradient and JVP computation in `JAX` without jitting the code. **Right:** Evolution of the memory footprint with de batch size

In figure 2.12, we plot the results we get with the `PyTorch` implementation. One observes that in all the cases, the forward-over-reverse mode consumes more memory than the reverse-over-forward mode. It is almost at the same level as reverse-over-reverse mode, which is quite unexpected.

The right plot of the evolution of the memory footprint with the batch size for the ResNet34 architecture evolves linearly, as expected.

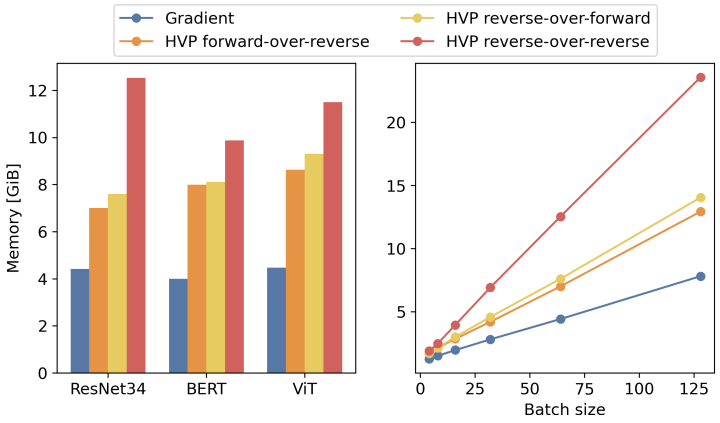


Figure 2.13: **Left:** Memory footprint of the gradient and JVP computation in `PyTorch`. **Right:** Evolution of the memory footprint with de batch size

CHAPTER 3

GRADIENT-BASED ALGORITHMS FOR BILEVEL OPTIMIZATION

Sommaire

This chapter reviews the standard approaches to solving bilevel optimization problems in the machine learning community that use gradient information.

3.1 Implicit differentiation

In this section, we consider the singleton lower-level problem presented in subsection 1.2.2:

$$\min_{x\in\mathbb{R}^{d_x}} \Phi(x) \triangleq f(x, y^*(x)) \quad \text{s.t.} \quad y^*(x) = \arg\min_{y\in\mathbb{R}^{d_y}} g(x,y) \ .$$

We give conditions on the outer and the inner functions to ensure that the value function Φ is differentiable and the main recipes in the bilevel optimizers that leverage the implicit differentiation technique we present.

3.1.1 The Implicit Function Theorem

The implicit function theorem (IFT) (Dontchev and Rockafellar, 2009; Krantz et al., 2013) is a fundamental result in analysis that enables the expression of the solution of some parametric equation as a differentiable function. We will use it later to derive the gradient of the value function in bilevel optimization.

Theorem 3.1. *Let $F : \mathbb{R}^p \times \mathbb{R}^d \to \mathbb{R}^p$ a continuously differentiable function. Let $(u_0, v_0) \in \mathbb{R}^p \times \mathbb{R}^p$ such that $F(u_0, v_0) = 0$ and the partial Jacobian $\partial_u F(u_0, v_0)$ is invertible. Then, there exists a neighborhood U of u_0 and a neighborhood V of v_0 and a continuously differentiable function $h : V \to U$ such that for any $v \in V$, it holds*

$$F(h(v), v) = 0 \ .$$

Moreover, the Jacobian of h is given by

$$\mathrm{d}h(v) = -\left[\partial_u F(h(v), v)\right]^{-1} \partial_v F(h(v), v) \ .$$

In the next section, we see how one can use the IFT to derive the gradient of the value function Φ in bilevel optimization.

3.1.2 Implicit differentiation and regularity of the value function

Gradient descent presented in Section 2.2 is the workhorse algorithm for differentiable optimization. The main advantage of first-order algorithms with respect to zero-order algorithms is that their guarantees are independent of the problem's dimension (Nesterov, 2018). However, to use gradient descent on the value function Φ, we must ensure that the function Φ is differentiable, and we must be able to compute its gradient efficiently. The value function Φ considered in bilevel optimization is defined implicitly. Therefore, it is not straightforward that these conditions hold. Indeed, the differentiability of the outer function f and the inner function g is not sufficient to make the function Φ differentiable, as shown in Example 3.1.

Example 3.1. *Consider the function f and g defined on $\mathbb{R}^2$ as*

$$f(x, y) = y \quad \textit{and} \quad g(x, y) = (y^3 - x)^2 \ .$$

Both f and g are differentiable with respect to (x, y). Moreover, for any $x \in \mathbb{R}$, the function $g(x, \cdot)$ admits a unique minimizer given by $y^*(x) = \mathrm{sign}(x)|x|^{\frac{1}{3}}$. However, the value function Φ given by

$$\Phi(x) = f(x, y^*(x)) = \mathrm{sign}(x)|x|^{\frac{1}{3}}$$

is not differentiable at $x = 0$ (see figure 3.1).

Consequently, one needs more assumptions to ensure the differentiability of the value function. This can be achieved by looking at the optimality condition that should verify $y^*(x)$ for any $x \in \mathbb{R}^{d_x}$ (Jongen et al., 1990; Dempe, 1993, 1998). In particular, as stated in Proposition 2.2, when $g(x, \cdot)$ is convex and has a unique minimizer, the minimizer $y^*(x)$ is uniquely defined by the solution in y of the equation

$$\nabla_y g(x, y) = 0 \ .$$

The Implicit Function Theorem (Theorem 3.1) enables to express the minimizer $y^*(x)$ as a differentiable function of x, under the condition that $\nabla_y g$ is differentiable and $\nabla^2_{yy} g(x, y^*(x))$ is invertible for any $x \in \mathbb{R}^{d_x}$. The common assumptions adopted to ensure these conditions are to assume that the inner function g is twice differentiable and that for any $x \in \mathbb{R}^{d_x}$, the function $g(x, \cdot)$ is μ_g-strongly convex (Pedregosa, 2016; Ghadimi and Wang, 2018) for some constant $\mu_g > 0$.

Proposition 3.1. *Assume that the function f is differentiable and that the inner function g is twice differentiable and μ_g-strongly convex with respect to y. Then, the value function Φ defined as in Problem (1.11) is differentiable and its gradient is given by*

$$\nabla\Phi(x) = \nabla_x f(x, y^*(x)) - \nabla_{xy} g(x, y^*(x)) \left[\nabla^2_{yy} g(x, y^*(x))\right]^{-1} \nabla_y f(x, y^*(x)) \ . \tag{3.1}$$

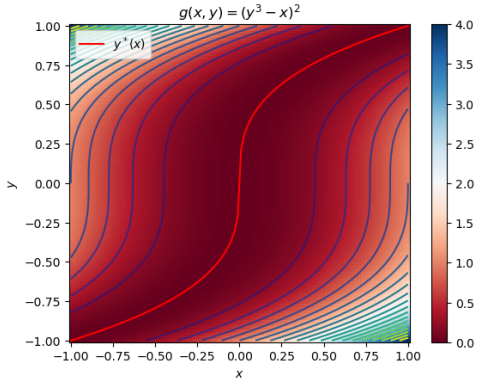


Figure 3.1: Example of differentiable function g where the function y^* is well-defined but not differentiable everywhere. Here, we have $g(x, y) = (y^3 - x)^2$ and $y^*(x) = \operatorname{sign}(x)|x|^{\frac{1}{3}}$. The blue lines are the level sets of g. The red curve is the graph of y^*. Near to $x = 0$, the function y^* has an infinite slope, making it non-differentiable at 0.

The gradient $\nabla\Phi$ is often called the hypergradient in the bilevel optimization literature.

Proof. Let $x \in \mathbb{R}^{d_x}$. Since the function $g(x, \cdot)$ is μ_g-strongly convex, it has a unique minimizer which is the solution of the equation

$$\nabla_y g(x, y) = 0 \ .$$

The function g is twice differentiable, and by strong convexity, its Hessian matrix $\nabla^2_{yy} g(x, y^*(x))$ is invertible. By the Implicit Function Theorem, there exists a differentiable function $y^* : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ such that for any $x \in \mathbb{R}^{d_x}$, it holds

$$\nabla_y g(x, y^*(x)) = 0 \ . \tag{3.2}$$

Therefore, the value function Φ is differentiable as a sum and composition of differentiable functions. Then, the chain rule yields

$$\nabla\Phi(x) = \nabla_x f(x, y^*(x)) + \mathrm{d}y^*(x)^\top \nabla_y f(x, y^*(x)) \tag{3.3}$$

where $\mathrm{d}y^*(x)$ denotes the Jacobian matrix of y^*. Differentiating Equation (3.2) with respect to x gives

$$\nabla_{yx} g(x, y^*(x)) + \nabla^2_{yy} g(x, y^*(x)) \mathrm{d}y^*(x) = 0 \ .$$

This yields the following expression for the Jacobian of y^*

$$\mathrm{d}y^*(x) = -\left[\nabla^2_{yy} g(x, y^*(x))\right]^{-1} \nabla_{yx} g(x, y^*(x)) \ . \tag{3.4}$$

Then, plugging Equation (3.4) into Equation (3.3) provides the result. □

For convenience, we denote $v^*(x)$ the quantity $v^*(x) = -\left[\nabla^2_{yy} g(x, y^*(x))\right]^{-1} \nabla_y f(x, y^*(x))$, so that

$$\nabla\Phi(x) = \nabla_x f(x, y^*(x)) + \nabla_{xy} g(x, y^*(x)) v^*(x) \ .$$

Previously, we saw that getting descent guarantees with the gradient descent algorithm requires ensuring the gradient of the value function Φ is Lipschitz continuous. It is possible to have this property by adding to the assumptions of Proposition 3.1, the Lipschitz continuity of the outer function f and the Lipschitz continuity of the Hessian of the inner function g, as proved by Ghadimi and Wang (2018).

Proposition 3.2 (Ghadimi and Wang (2018, Lemma 2.2))**.** *Assume that the outer function f is $L_{f,1}$-smooth, $L_{f,0}$-Lipschitz continuous and that the inner function g is $L_{g,1}$-smooth and its Hessian is $L_{g,2}$-Lipschitz continuous. Moreover, assume that for any $x \in \mathbb{R}^{d_x}$, the function $g(x, \cdot)$ is μ_g-strongly convex. Then, the gradient of the value function Φ is L_Φ-Lipschitz continuous with*

$$L_\Phi = L_{f,1} + \frac{2L_{f,1}L_{g,2} + L^2_{f,0}L_{g,2}}{\mu_g} + \frac{2L_{f,0}L_{g,1}L_{g,2} + L_{f,1}(L_{g,1})^2}{\mu_g^2} + \frac{L_{f,0}(L_{g,1})^2 L_{g,2}}{\mu_g^3} \ .$$

Remark. The condition on the Lipschitz continuity of the outer function can be relaxed by a slightly weaker assumption, which consists of assuming the existence of a constant $C > 0$ such that for any $x \in \mathbb{R}^{d_x}$ we have

$$\|\nabla f(x, y^*(x))\| \leq C \ .$$

For the remainder of this section, we assume the following conditions, which are sufficient for the smoothness of the value function Φ.

Assumption 3.1. *The outer function f is differentiable, $L_{f,0}$-Lipschitz continuous and $L_{f,1}$-smooth.*

Assumption 3.2. *The inner function g is twice differentiable, $L_{g,1}$-smooth, and its second order derivative is $L_{g,2}$-Lipschitz continuous. Moreover, for any $x \in \mathbb{R}^{d_x}$, the function $g(x, .)$ is μ_g-strongly convex.*

For instance, the strong convexity of the inner function g holds when g is the Ridge regression loss or the ℓ^2-regularized logistic regression loss with non-separable data. Although the uniformity of the strong convexity modulus μ_g for all $x \in \mathbb{R}^{d_x}$ is very restrictive, it is a common assumption in the bilevel literature (Ji et al., 2021; Grazzi et al., 2020).

One limitation of implicit differentiation is that it requires the strong convexity of the inner function, which is very restrictive. In practical applications involving neural networks, for instance, the inner function can be non-convex. Petrulionyte et al. (2024) propose to circumvent this non-convexity by replacing the inner problem it with an infinite-dimensional problem. Specifically, in machine learning, the inner function often takes the form

$$g(x, y) = \mathcal{L}(x, h_y)$$

where $(h_y)_{y \in \mathbb{R}^{d_y}}$ is a parametric family of functions (e.g. a neural network parametrized by its weights). Often, the non-convexity of the problem comes from the parametrization, meaning that function $\mathcal{L}(x, \cdot)$ is convex in many cases. Thus, the idea of Petrulionyte et al. (2024) is to replace the minimization of g with respect to y by the minimization of the functional $\mathcal{L}(x, \cdot)$ in functional space and demonstrate a functional variant of the implicit function theorem that enables to get the Frechet derivative of $h^*(x) = \arg\min_h \mathcal{L}(x, h)$.

In other machine learning applications, the inner function is nonsmooth. This is the case, for instance, when using nonsmooth regularization to enforce some sparsity pattern to the inner problem, as in Lasso regression (Tibshirani, 1996). Bolte et al. (2021) provide a nonsmooth adaptation of the implicit function theorem for functions that are path differentiable (Bolte and Pauwels, 2020). Grazzi et al. (2024) extended these by providing convergence rates for algorithms leveraging nonsmooth implicit differentiation. Another line of work (Bertrand et al., 2020, 2022) proposes to differentiate optimality conditions for composite inner functions.

In the remainder of the thesis, we focus on cases where the inner function is smooth. The next subsection provides details of the practical use of implicit differentiation to solve bilevel problems.

3.1.3 Approximate Implicit Differentiation (AID)

The gradient expression given in Equation (3.1) presents two computational bottlenecks: it requires the resolution of the inner problem to get $y^*(x)$ and the resolution of a linear system to get $v^*(x)$. When the inner problem is ill-conditioned, this can be dramatically costly. In its seminal work, Pedregosa (2016) lays the first bricks to Approximate Implicit Differentiation (AID), opening the doors to many algorithms for gradient-based bilevel optimization. The idea is the following: instead of computing the gradient $\nabla\Phi(x)$ exactly by solving the two subproblems, one can replace $y^*(x)$ and $v^*(x)$ in Equation (3.1) by approximations y and v. This leads us to consider the approximate hypergradient

$$\bar{\nabla}\Phi(x; y, v) = \nabla_x f(x, y) + \nabla^2_{xy} g(x, y) v \ . \tag{3.5}$$

Algorithm 4 General AID-based algorithm

Input: initializations $x^0 \in \mathbb{R}^{d_x}$, $y_0 \in \mathbb{R}^{d_y}$, $v^0 \in \mathbb{R}^{d_y}$, number of iterations T
for $t = 0, \dots, T-1$ **do**
Get an approximation y^{t+1} of $y^*(x^t)$
Get an approximation v^{t+1} of the solution of the linear system

$$\nabla^2_{yy} g(x^t, y^{t+1}) v = -\nabla_y f(x^t, y^{t+1})$$

Update x following the opposite direction of the approximate gradient

$$x^{t+1} = x^t - \gamma \bar{\nabla} \Phi(x^t; y^{t+1}, v^{t+1})$$

end for

This approximate hypergradient is consistent with the exact hypergradient since we have

$$\bar{\nabla}\Phi(x; y^*(x), v^*(x)) = \nabla \Phi(x) \ .$$

The approximations y and v are the outputs of iterative algorithms that approximately solve the inner problem and the linear system. Running gradient descent with the approximate gradient $\bar{\nabla}\Phi(x; y, v)$ yields the AID-based algorithms, whose general form is given in Algorithm 4. At iteration t, AID-based algorithm computes y^{t+1}, an approximation of $y^*(x^t)$. Then, one needs an approximation of $v^*(x^t)$. However, since we do not have $y^*(x^t)$, we rather approximate

$$-\nabla^2_{yy} g(x^t, y^{t+1}) \nabla_y f(x^t, y^{t+1}) \ .$$

We thus denote $\hat{v}(x; y) \triangleq -\nabla^2_{yy} g(x, y) \nabla_y f(x, y)$, so that $\hat{v}(x; y^*(x)) = v^*(x)$.

Pedregosa (2016) proposes an instantiation of Algorithm 4 which consists in solving at iteration t the subproblems with a precision ϵ^t, for a given tolerance sequence $(\epsilon^t)_{t \geq 0}$. He shows that if the decrease of the sequence $(\epsilon^t)_{t \geq 0}$ is sufficient, the gradient norms $(\|\nabla \Phi(x^t)\|)_{t \geq 0}$ converge towards zero.

Theorem 3.2 (Pedregosa (2016, Theorem 2))**.** *Assume that f has Lipschitz gradients, g is strongly convex with respect to y, and has Lipschitz gradients and Hessian. If the tolerance sequence $(\epsilon^t)_{t \geq 0}$ verifies*

$$\sum_{t=0}^{+\infty} \epsilon^t < +\infty$$

then we have

$$\lim_{t \to +\infty} \|\nabla \Phi(x^t)\| = 0 \ .$$

This first theoretical result provides a sufficient condition for converging with an AID method. However, it is an asymptotic result that does not give any convergence speed. The first non-asymptotic convergence result for AID-based algorithms is due to Ghadimi and Wang (2018).

It is interesting to see how the gradient error influences the descent property of the AID-based algorithms. This is done in Lemma 3.1 by leveraging the smoothness of the value function Φ.

Lemma 3.1. *Assume that Assumptions 3.1 and 3.2 hold. Let $(y^t, v^t, x^t)_{t \geq 0}$ a sequence of iterates produced by Algorithm 4. Then, we have the following descent property*

$$\begin{aligned} \Phi(x^{t+1}) \leq \Phi(x^t) - \frac{\gamma}{2} \|\nabla \Phi(x^t)\|^2 - \frac{\gamma}{2} \left(1 - L_\Phi \gamma\right) \|\bar{\nabla}\Phi(x^t; y^{t+1}, v^{t+1})\|^2 \\ + \frac{\gamma}{2} \|\nabla \Phi(x^t) - \bar{\nabla}\Phi(x^t; y^{t+1}, v^{t+1})\|^2 \end{aligned} \tag{3.6}$$

Proof. Using the smoothness of the value function Φ in Proposition 3.2, we have the following inequality for any $x, x' \in \mathbb{R}^{d_x}$

$$\Phi(x') \leq \Phi(x) + \langle \nabla\Phi(x), x' - x\rangle + \frac{L_\Phi}{2}\|x' - x\|^2 \ .$$

Using the previous inequality with $x = x^t$ and $x' = x^{t+1}$ and the relation

$$x^{t+1} - x^t = -\gamma\bar{\nabla}\Phi(x^t; y^{t+1}, v^{t+1})$$

we have

$$\Phi(x^{t+1}) \leq \Phi(x^t) - \gamma\langle \nabla\Phi(x^t), \bar{\nabla}\Phi(x^t; y^{t+1}, v^{t+1})\rangle + \frac{\gamma^2 L_\Phi}{2}\|\bar{\nabla}\Phi(x^t; y^{t+1}, v^{t+1})\|^2 \ .$$

For any $a, b \in \mathbb{R}^{d_x}$, we have $\langle a, b\rangle = \frac{1}{2}(\|a\|^2 + \|b\|^2 - \|a - b\|^2)$. Applying this relation and rearranging yields

$$\begin{aligned}\Phi(x^{t+1}) \leq \Phi(x^t) - \frac{\gamma}{2}\|\nabla\Phi(x^t)\|^2 - \frac{\gamma}{2}\left(1 - L_\Phi\gamma\right)\|\bar{\nabla}\Phi(x^t; y^{t+1}, v^{t+1})\|^2 \\ + \frac{\gamma}{2}\|\nabla\Phi(x^t) - \bar{\nabla}\Phi(x^t; y^{t+1}, v^{t+1})\|^2\end{aligned}$$

□

In the case where $y^{t+1} = y^*(x^t)$, $v^{t+1} = v^*(x^t)$ and $\gamma \leq \frac{1}{L_\Phi}$, we recover the usual descent lemma for gradient descent for smooth functions (Lemma 2.1). Moreover, one might note that the approximation error term $\frac{\gamma}{2}\|\nabla\Phi(x^t) - \bar{\nabla}\Phi(x^t; y^{t+1}, v^{t+1})\|^2$ slows down the descent. In other words, the better the approximation of the hypergradient, the steepest the descent. One can bound this approximation error by the error on y and v.

Lemma 3.2. *Assume that Assumptions 3.1 and 3.2. For $x \in \mathbb{R}^{d_x}$, and $y, v \in \mathbb{R}^{d_y}$, we have*

$$\|\nabla\Phi(x) - \bar{\nabla}\Phi(x; y, v)\| \leq \bar{L}(\|y - y^*(x)\| + \|v - v^*(x)\|) \ .$$

As a consequence, the Equation (3.6) can be rewritten as

$$\begin{aligned}\Phi(x^{t+1}) \leq \Phi(x^t) - \frac{\gamma}{2}\|\nabla\Phi(x^t)\|^2 - \frac{\gamma}{2}\left(1 - L_\Phi\gamma\right)\|\bar{\nabla}\Phi(x^t; y^{t+1}, v^{t+1})\|^2 \\ + \frac{\gamma^2\bar{L}^2}{2}\left(\|y^{t+1} - y^*(x^t)\|^2 + \|v^{t+1} - v^*(x^t)\|^2\right)\end{aligned} \tag{3.7}$$

Proof. Let $(y, v, x) \in \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \times \mathbb{R}^{d_x}$. We start by writing

$$\begin{aligned}\|\bar{\nabla}\Phi(x; y, v) - \nabla\Phi(x)\| &\leq \|\nabla_x f(x, y) - \nabla_x f(x, y^*(x))\| + \|\nabla^2_{xy} g(x, y)v - \nabla^2_{xy} g(x, y^*(x))v^*(x)\| \\ &\leq \|\nabla_x f(x, y) - \nabla_x f(x, y^*(x))\| + \|\nabla^2_{xy} g(x, y)\|\|v - v^*(x)\| \\ &\quad + \|v^*(x)\|\|\nabla^2_{xy} g(x, y) - \nabla^2_{xy} g(x, y^*(x))\| \ .\end{aligned}$$

We bound the first term using the $L_{f,1}$-Lipschitz continuity $\nabla_x f$. For the second term, we use the boundedness of $\nabla^2_{xy} g$ thanks to the Lipschitz continuity of $\nabla_y g(\,\cdot\,, y)$. For the third term, we use that $\nabla^2_{xy} g(x, \,\cdot\,)$ is $L_{g,2}$-Lipschitz continuous. We finally get

$$\begin{aligned}\|\bar{\nabla}\Phi(x; y, v) - \nabla\Phi(x)\| &\leq L_{f,1}\|y - y^*(x)\| + L_{g,1}\|v - v^*(x)\| + \frac{L_{f,0}L_{g,2}}{\mu_g}\|v - v^*(x)\| \\ &\leq \left(L_{f,1} + \frac{L_{f,0}L_{g,2}}{\mu_g}\right)\|y - y^*(x)\| + L_{g,1}\|v - v^*(x)\| \ .\end{aligned}$$

Taking $\bar{L} = \sqrt{2}\max\left(L_{f,1} + \frac{L_{f,0}L_{g,2}}{\mu_G}, L_{g,1}\right)$ yields

$$\|\bar{\nabla}\Phi(x; y, v) - \nabla\Phi(x)\|^2 \leq \bar{L}^2(\|y - y^*(x)\|^2 + \|v - v^*(x)\|^2) \ . \tag{3.8}$$

□

The combination of Lemma 3.1 and Lemma 3.2 suggests that the efficiency of AID-based algorithms depends on the quality of the approximations of $y^*(x)$ and $v^*(x)$ we use. However, getting better approximations of $y^*(x)$ and $v^*(x)$ demands more computations leading to heavier outer iterations. As a consequence, there is a trade-off between the convergence speed in terms of number of outer iterations and the computational cost of each outer iteration.

Besides being able to develop gradient-based methods, having the function Φ differentiable enables a criterion to compare the theoretical performances of the bilevel solvers. In the bilevel literature, the criterion given by Definition 3.1 is often used.

Definition 3.1. Let $(x^t)_{0\leq t\leq T-1}$ the iterates of an algorithm. We say that we have found an ϵ-stationary point if

$$\frac{1}{T}\sum_{t=1}^{T-1}\mathbb{E}[\|\nabla\Phi(x^t)\|^2]\leq\epsilon$$

where the expectation is taken with respect to the possible randomness of the algorithm producing the sequence $(x^t)_{0\leq t\leq T-1}$.

In the next subsection, we review some of the main strategies deployed to approximate $y^*(x^t)$ and $v^*(x^t)$.

3.1.4 Practical computation of the approximate hypergradient

Resolution of the inner problem. Since the inner function is assumed to be strongly convex, one can approximate efficiently the solution of the inner problem by performing several steps of a linearly convergent optimization algorithm. The solver most widely used in the bilevel literature is the gradient descent presented in Chapter 2 (Ghadimi and Wang, 2018; Ji et al., 2021). Ghadimi and Wang (2018),Ji and Liang (2023) and Chen et al. (2023b) propose to use gradient descent with Nesterov acceleration (Nesterov, 1983), that enjoy a faster convergence speed on strongly convex functions. Pedregosa (2016) proposes to use L-BFGS (Liu and Nocedal, 1989), a quasi-Newton method for the inner problem.

Resolution of the linear system. The hypergradient given in Equation (3.1) involved a linear system driven by the Hessian matrix of the inner function g. In the literature, several authors (Pedregosa, 2016; Grazzi et al., 2020) propose to perform several steps of Conjugate Gradient algorithm (Hestenes and Stiefel, 1952; Nocedal and Wright, 2006) to approximate the solution of the linear system. In Ramzi et al. (2022), the authors propose an efficient method to approximate the hypergradient Equation (3.1) when the inner problem is solved by a quasi-Newton method (Liu and Nocedal, 1989; Broyden, 1965). Indeed, as the quasi-Newton techniques rely on an approximation of the inverse Hessian of the function we want to minimize, the authors propose to reuse the quasi-Newton matrices of the forward pass for the approximation of the solution of the linear system. Other authors, such as Grazzi et al. (2020) or Arbel and Mairal (2022a), cast the linear system as the minimization of the following quadratic function

$$v\mapsto\frac{1}{2}v^\top\nabla^2_{yy}g(x,y)v+\nabla_y f(x,y)^\top v$$

and apply gradient descent steps to it. This is justified by the Hessian matrix $\nabla^2_{yy}g(x,y)$ being symmetric positive definite since $g(x,\cdot)$ is strongly convex.

Choosing which solver to use for each subproblem is not the only question. Whether for the resolution of the inner problem or the resolution of the linear system, one has to select the initialization of each sub-procedure. In the next section, we present the two main strategies for this initialization: the cold-start and the warm-start strategies.

3.1.5 Initialization of the sub-procedures

An important question when designing AID-based algorithms is the choice of the initialization of the two sub-procedures that approximate $y^*(x)$ and $v^*(x)$. There are two main strategies: the cold-start

and the warm-start strategies. Both methods yield different computational complexities (Arbel and Mairal, 2022a; Ji et al., 2021) but also different implicit biases in overparametrized cases (Vicol et al., 2022). In what follows, we focus on the computational aspects of the choice of initialization.

For this subsection, we denote for $k \in \mathbb{N}$ $\mathcal{A}_k(y; x)$ the kth-iterate of an iterative algorithm that approximates $y^*(x)$, starting from $y \in \mathbb{R}^{d_y}$. Similarly, we denote $\mathcal{B}_q(v; y, x)$ the qth-iterate of an iterative algorithm that approximates $\hat{v}(x; y)$, starting from $v \in \mathbb{R}^{d_y}$.

Cold-start strategy

The cold-start strategy uses the same initialization for the subproblems at each outer iteration. In other words, for $y^0, v^0 \in \mathbb{R}^{d_y}$, we take at iteration t

$$y^{t+1} = \mathcal{A}_{k_t}(y^0; x^t), \quad v^{t+1} = \mathcal{B}_{q_t}(v^0; y^{t+1}, x^t) \ .$$

In Ghadimi and Wang (2018), the authors propose the algorithm BA, an instantiation of Algorithm 4 where the inner problem is solved by gradient steps and cold started. The linear system is assumed to be solved exactly. They show that under Assumption 3.1 and Assumption 3.2 and by furthermore assuming that $\sup_{x \in \mathbb{R}^{d_x}} \|y^0 - y^*(x)\|$ is finite, the iterates of the BA algorithm converge *if we choose a number of inner steps* $k_t = \Theta(\sqrt[4]{t+1})$. This growing number of inner iterations with the outer iterations is mandatory to reduce the hypergradient gradient approximation error.

In Grazzi et al. (2023), the authors show that with the cold-start strategy, an instantiation of Algorithm 4 where gradient descent is used for the inner and the linear system solvers, one can recover the convergence speed of gradient descent for non-convex single-level problems. This is proved under the same assumptions as for the BA algorithm. To do so, they assume the number of iterations of both subprocedures verifies $k_t = q_t = \Theta(\log(t))$. Once again, the theory suggests a growing number of inner iterations with the outer iterations.

To illustrate the behavior of bilevel optimizers with cold-started subproblem solvers, we consider the following synthetic problem where the inner and the outer functions are quadratics

$$f(x, y) = \frac{1}{2} x^\top A_{f,x} x + b_{f,x}^\top x + \frac{1}{2} y^\top A_{f,y} y + b_{f,y}^\top y + x^\top B_f y \tag{3.9}$$

$$g(x, y) = \frac{1}{2} x^\top A_{g,x} x + b_{g,x}^\top x + \frac{1}{2} y^\top A_{g,y} y + b_{g,y}^\top y + x^\top B_g y \tag{3.10}$$

with $A_{f,x}$, $A_{g,x} \in \mathbb{R}^{10\times 10}$, $b_{f,x}$, $b_{g,x} \in \mathbb{R}^{10}$, $A_{f,y}$, $A_{g,y} \in \mathbb{R}^{100\times 100}$, $b_{f,y}$, $b_{g,y} \in \mathbb{R}^{100}$, B_f, $B_g \in \mathbb{R}^{10\times 100}$. We run the Algorithm 4 with the inner solver (gradient descent) cold started at each iteration for different values of number of inner steps. The linear system is solved exactly with the conjugate gradient algorithm (which is still doable at this scale). We report in figure 3.2 the gradient norm of the value function with respect to wall-clock time. We observe that in all the cases, the gradient norm reaches a plateau. Moreover, the higher the number of inner steps, the lower the plateau. But, can also notice that the computing time increases with the number of inner steps, which is expected.

Warm-start strategy

The warm-starting strategy consists of initializing each subproblem with the approximate solutions used in the previous outer iteration. Formally, this means that we take

$$y^{t+1} = \mathcal{A}_{k_t}(y^t; x^t), \quad v^{t+1} = \mathcal{B}_{q_t}(v^t; y^{t+1}, x^t) \ .$$

This strategy was first proposed by Ji et al. (2021) with the AID-BiO algorithm, where the inner solution is approached by gradient descent steps, and the solution of the linear system is approximated by the conjugate gradient algorithm. The authors show that under Assumption 3.1 and Assumption 3.2, the complexity of AID-BiO to find an ϵ-stationary point of Φ is $\mathcal{O}(\epsilon^{-1})$. Notably, this result assumes a fixed

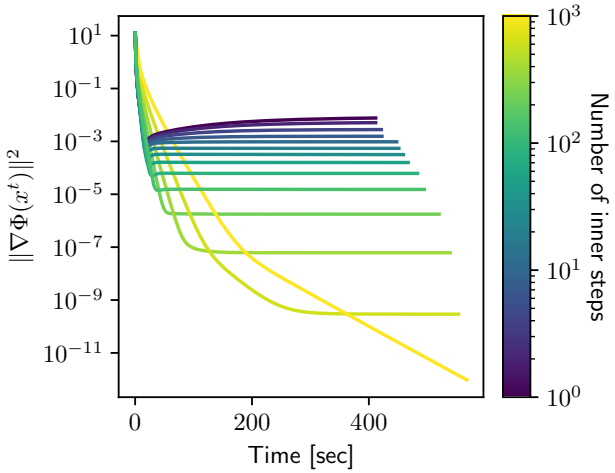


Figure 3.2: Gradient norm of the value function with respect to wall-clock time when the outer and the inner functions are quadratics. On the one hand, the more inner steps at each outer iteration, the lower the plateau of the gradient norm. On the other hand, the more inner steps, the higher the computing time.

number of steps of the inner problem and the linear system solvers. Also, it does not require to assume that $\sup_{x\in\mathbb{R}^{d_x}} \|y^0 - y^*(x)\|$ is finite. In this paper, they also introduce the warm-starting strategy for stochastic AID-based algorithms with stocBiO. However, only the inner solver is warm-started. Arbel and Mairal (2022a) built upon by proposing AmIGO, the first stochastic AID-based algorithm in which the inner and the linear system solvers are cold started. A more in-depth presentation of stochastic AID-based algorithms is presented in Section 3.2.

Finally, the warm-start strategy presents a drawback: it requires storing the last iterates of the two subprocedures. This memory burden prevents this technique from being used in multi-block scenarios like the deep equilibrium models' training because it would require storing one inner variable per sample.

To understand how the initialization strategy influences the convergence of the bilevel algorithms, we run an instantiation of Algorithm 4 where at each iteration, 10 steps of gradient descent on the inner problem and on the linear system are performed. We consider the quadratic setting of Equation (3.9) and Equation (3.10). We do four runs depending we warm-start or cold start-each each subsolver. We report in figure 3.3 the gradient norm of the value function with respect to wall-clock time. The observation we make is that warm-starting both subsolvers enables the convergence of the gradient norm towards zero. Otherwise, the gradient norm reaches a plateau.

So far, we have focused on deterministic algorithms that implement approximate implicit differentiation. In many machine learning scenarios, these algorithms do not scale with the sample size of the considered problem. This has motivated the development of stochastic algorithms for bilevel optimization, which we present in the next section.

3.2 Stochastic Approximate Implicit Differentiation

In many machine learning applications like those presented in Section 1.3, the inner function and/or the outer function take the form of an expectation over distributions $\mathbb{P}_g$ and $\mathbb{P}_f$ respectively

$$f(x,y) = \mathbb{E}_{\xi\sim\mathbb{P}_f}[f(x,y;\xi)] \quad \text{and} \quad g(x,y) = \mathbb{E}_{\zeta\sim\mathbb{P}_g}[g(x,y;\zeta)] \ . \tag{3.11}$$

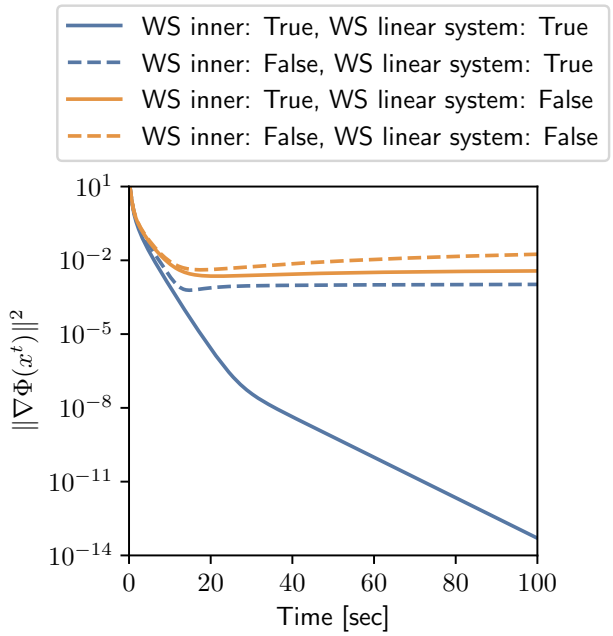


Figure 3.3: Gradient norm of the value function in function with respect to wall-clock time. Solid lines correspond to runs with warm-started inner problems while dashed lines correspond to runs with cold-started inner problems. Blue lines correspond to runs with warm-started linear system resolution while orange lines correspond to runs with cold-started linear system resolution. In the legend, "WS" means "warm-start"

A classical special case of this setting is the empirical risk minimization problem, where the functions f and g are empirical means over samples:

$$f(x,y) = \frac{1}{m}\sum_{j=1}^{m} f_j(x,y) \quad \text{and} \quad g(x,y) = \frac{1}{n}\sum_{i=1}^{n} g_i(x,y) \ . \tag{3.12}$$

It corresponds to the case where ξ and ζ are uniform random variables over $[m]$ and $[n]$ respectively. For instance, in the hyperparameter optimization problem, the outer function f is the average of the loss over the validation sample, and the inner function g is the average of the loss over the training sample. For convenience, for two batches of realizations of ξ and ζ $\mathcal{B}_f = \{\xi, \dots, \xi_b\}$ and $\mathcal{B}_g = \{\zeta_1, \dots, \zeta_b\}$, we denote

$$f_{\mathcal{B}_f}(x,y) = \frac{1}{b}\sum_{j=1}^{b} f(x,y;\xi_j) \quad \text{and} \quad g_{\mathcal{B}_g}(x,y) = \frac{1}{b}\sum_{i=1}^{b} g_i(x,y;\zeta_i) \ . \tag{3.13}$$

In single-level optimization, classical stochastic algorithms such as SGD rely on an unbiased estimation of the gradient of the objective function. In the bilevel case, the gradient of the value function Φ is given by Equation (3.1). Notably, it involves the inverse Hessian-vector product (iHVP) $v^*(x)$ which uses the inverse of the Hessian matrix of the inner function g. This matrix inversion challenges the design of an unbiased estimator of the hypergradient since in particular, we have

$$\mathbb{E}_{\zeta\sim\mathbb{P}_g}\left[\nabla^2_{yy} g(x,y;\zeta)^{-1}\right] \neq \left[\mathbb{E}_{\zeta\sim\mathbb{P}_g}\left[\nabla^2_{yy} g(x,y;\zeta)\right]\right]^{-1} \ .$$

However, many works have proposed different ways to handle stochasticity, notably by using stochastic algorithms for the two subprocedures, leading to different performances. In addition to the regularity Assumptions 3.1 and 3.2, these works assume the unbiasedness and the bounded variance of the different stochastic derivatives (Ghadimi and Wang, 2018; Chen et al., 2021)

Assumption 3.3. *The stochastic derivatives* $\nabla_x f(x,y;\xi)$, $\nabla_y f(x,y;\xi)$, $\nabla_y g(x,y;\zeta)$, $\nabla^2_{yy} g(x,y;\zeta)$ *and* $\nabla^2_{xy} g(x,y;\zeta)$ *are respectively unbiased estimators of* $\nabla_x f(x,y)$, $\nabla_y f(x,y)$, $\nabla_y g(x,y)$, $\nabla^2_{yy} g(x,y)$ *and*

$\nabla^2_{xy} g(x,y)$. *Moreover, there exist* σ_f *and* σ_g *such that*

$$\begin{aligned}
\mathbb{E}_{\xi\sim\mathbb{P}_f}[\|\nabla_x f(x,y;\xi) - \nabla_x f(x,y)\|^2] &\leq \sigma_f^2 \\
\mathbb{E}_{\xi\sim\mathbb{P}_f}[\|\nabla_y f(x,y;\xi) - \nabla_y f(x,y)\|^2] &\leq \sigma_f^2 \\
\mathbb{E}_{\zeta\sim\mathbb{P}_g}[\|\nabla_y g(x,y;\zeta) - \nabla_y g(x,y)\|^2] &\leq \sigma_g^2 \\
\mathbb{E}_{\zeta\sim\mathbb{P}_g}[\|\nabla^2_{yy} g(x,y;\zeta) - \nabla_y g(x,y)\|^2] &\leq \sigma_g^2 \\
\mathbb{E}_{\zeta\sim\mathbb{P}_g}[\|\nabla^2_{xy} g(x,y;\zeta) - \nabla_y g(x,y)\|^2] &\leq \sigma_g^2 \quad .
\end{aligned}$$

In some papers (Ji et al., 2021; Yang et al., 2021), the regularity assumptions on the outer function (Assumption 3.1) and on the inner function (Assumption 3.2) are assumed to hold on the stochastic oracles, which is a bit stronger.

Assumption 3.4. *The function* $f(\cdot\,,\cdot\,;\xi)$ *is differentiable,* $L_{f,0}$*-Lipschitz continuous and* $L_{f,1}$*-smooth.*

Assumption 3.5. *The function* $g(\cdot\,,\cdot\,;\zeta)$ *is twice differentiable,* $L_{g,1}$*-smooth, and its second order derivative is* $L_{g,2}$*-Lipschitz continuous. Moreover, for any* $x \in \mathbb{R}^{d_x}$*, the function* $g(x,.;\zeta)$ *is* μ_g*-strongly convex.*

In the next subsections, we review the methods that are used in the stochastic AID-based bilevel solvers to approximate $y^*(x^t)$ and $v^*(x^t)$ in a stochastic fashion and present the guarantees we get for those methods.

3.2.1 Two-loop versus one-loop algorithms

Analogously to the deterministic case, the common strategy to approximate the inner solution is to run one or several steps of a stochastic optimization algorithm. Thus, the first stochastic AID-based algorithm is the Bilevel Stochastic Algorithm (BSA) (Ghadimi and Wang, 2018) where the solution of the inner problem is approximated by several SGD steps. Since the solver is cold-started, the analysis of BSA assumes a number of inner steps equal to $\lceil\sqrt{t+1}\rceil$ where t is the current outer iteration number. Subsequent works (Ji et al., 2021; Arbel and Mairal, 2022a; Chen et al., 2021) also use SGD for the inner problem but with warm-start, which enables to keep the number of inner steps fixed on the order of the inner conditioning, that $\frac{L_{g,1}}{\mu_g} \geq 1$.

In Hong et al. (2023), the authors propose a novel paradigm, where only a single step is performed for the inner problem. The practical advantage of this is that the practitioners do not need to tune the number of inner steps. However, the analysis of their method, named Two-Time Scale Algorithm (TTSA), assumes an inner step size in $\Theta(T^{-\frac{3}{5}})$ and an outer step size in $\Theta(T^{-\frac{2}{5}})$. In other words, for large horizons, the progress of the inner variable is much faster than the progress of the outer variable. This lead to complexity in $\tilde{\mathcal{O}}(\epsilon^{-\frac{5}{2}})$ (here the tilde on the $\mathcal{O}$ notation indicates a hidden $\log(\epsilon^{-1})$ factor), which is far from the complexity of SGD for non-convex problems, which is in $\mathcal{O}(\epsilon^{-2})$. Chen et al. (2022b) propose STABLE, a single-loop algorithm where the inner variable y is updated as follows

$$y^{t+1} = y^t - \rho\nabla g(x^t, y^t; \zeta_t) - [H^t_{yy}]^{-1} H^t_{yx}(x^{t+1} - x^t)$$

where H^t_{yy} and H^t_{yx} are stochastic estimators of $\nabla^2_{yy} g(x^t, y^t)$ and $\nabla^2_{yx} g(x^t, y^t)$ respectively. This dynamic motivated by an ODE analysis allows to get a complexity in $\mathcal{O}(\epsilon^{-2})$, but the computational cost of the product $[H^t_{yy}]^{-1} H^t_{yx}$ makes this algorithm impractical.

3.2.2 Variance reduction in bilevel optimization

Beyond the distinction between single-loop and two-loop algorithms, some papers propose to accelerate the problem resolution by implementing variance reduction techniques, either on the inner update or in the outer update. For instance, Yang et al. (2021) and Khanduri et al. (2021) propose to adapt the momentum-based variance reduction of the STORM algorithm (Cutkosky and Orabona, 2019).

This technique works as follows. Assume that w is the variable we want to update (w can be either the inner variable y, the linear system variable v, or the outer variable x) in a direction $F(w) = \mathbb{E}_\zeta[F(w;\zeta)]$ (F can be the inner gradient for instance). Then the STORM update rule reads

$$w^{t+1} = w^t - \rho^t d^t$$

where ρ^t is the step size and d^t verifies the following recursion

$$d^t = F(w^t;\zeta_t) + (1-\beta)(d^{t-1} - F(w^{t-1};\zeta_t))$$

with $\beta \in (0,1)$ a momentum parameter and $\zeta_t \sim \mathbb{P}_g$. Note that d^t is actually a biased estimate of $F(w^t;\zeta_t)$ since

$$\mathbb{E}_{\zeta_t}[d^t] = F(w^t) + (1-\beta)(d^{t-1} - F(w^{t-1})) \neq F(w^t) \ .$$

By implementing this variance reduction technique for the inner and the outer variables updates, Yang et al. (2021) and Khanduri et al. (2021) prove that their respective algorithms MRBO and SUSTAIN achieve a complexity in $\tilde{\mathcal{O}}(\epsilon^{-\frac{2}{3}})$.

Yang et al. (2021) also propose to use another variance reduction technique called SPIDER (Fang et al., 2018), or also knows as SARAH (Nguyen et al., 2017). As STORM, this technique is based on a recursive estimation of the update direction. However, the estimate direction is reset periodically, leading to a two-loop algorithm. At iteration t, the estimate direction $d^{t,0}$ is initialized by a full-batch (in case of finite sums) computation or a large batch estimate of $F(w^{t,0})$. Then, in an inner loop, the estimate direction is updated as follows

$$d^{t,k+1} = d^{t,k} + (F(w^{t,k};\zeta_{t,k}) - F(w^{t,k-1};\zeta_{t,k}))$$

and the variable w is updated as

$$w^{t,k+1} = w^{t,k} - \rho^{t,k} d^{t,k} \ .$$

This method also achieves a complexity in $\mathcal{O}(\epsilon^{-\frac{2}{3}})$.

In what follows, we explain how the iHVP is approximated in the stochastic AID-based algorithms.

3.2.3 Solving the linear system

As mentioned earlier, the hypergradient given in Equation (3.1) involves the following linear system

$$\mathbb{E}_{\zeta\sim\mathbb{P}_g}\left[\nabla^2_{yy} g(x,y;\zeta)\right] v = -\mathbb{E}_{\xi\sim\mathbb{P}_f}\left[\nabla_y f(x,y;\xi)\right] \ .$$

In the stochastic AID-based literature, mainly two methods have been proposed to estimate the solution of this linear system: one based on Neumann approximations and another based on the reformulation of the linear system resolution as a stochastic optimization problem.

Neumann approximations. A common technique to estimate the iHVP is to leverage the identity

$$A^{-1} = \sum_{k=0}^{+\infty} (I - A)^k$$

that holds for any square matrix A such that $\|A\| < 1$. Thus, for any $y \in \mathbb{R}^{d_y}$ and $x \in \mathbb{R}^{d_x}$, we can express the iHVP $\nabla^2_{yy} g(x,y)^{-1} \nabla f(x,y)$ as

$$\nabla^2_{yy} g(x,y)^{-1} \nabla f(x,y) = \eta \left[\sum_{k=0}^{+\infty} (I - \frac{1}{\eta}\nabla^2_{yy} g(x,y))^k\right] \nabla f(x,y)$$

where $\eta > 0$ is a constant such that $\eta > \|\nabla^2_{yy} g(x,y)\|$.

For a given $k > 0$, the term $(I - \frac{1}{\eta}\nabla^2_{yy} g(x,y))^k$ can be estimated by $\prod_{i=0}^{k-1}[I - \frac{1}{\eta}\nabla^2_{yy} g(x,y;\zeta_i)]$ where $\zeta_1, \dots, \zeta_{k-1}$ are drawn from $\mathbb{P}_g$.

In the context of bilevel optimization, this technique was first proposed by Ghadimi and Wang (2018) who suggest the following approximation of $\left[\nabla^2_{yy} g(x,y)\right]^{-1}$ for i.i.d. samples $\zeta_1, \dots, \zeta_b$ drawn from $\mathbb{P}_g$:

$$H_{\text{hia}} = \eta Q \prod_{q=1}^{p} [I - \eta \nabla^2_{yy} g(x, y; \zeta_q)] \tag{3.14}$$

where $\eta \in \left(0, \frac{1}{L_{g,1}}\right)$, $Q \in \mathbb{N}$ and p is a random variable uniformly distributed in $\{0, \dots, Q-1\}$. This estimation comes with the guarantee in Proposition 3.3.

Proposition 3.3 (Ghadimi and Wang (2018, Lemma 3.2))**.** *Assume that Assumption 3.2 and Assumption 3.3 hold. Let H_{hia} defined in Equation (3.14). Then by taking $\eta \leq \frac{1}{L_{g,1}}$, it holds*

$$\begin{aligned} \| \left[\nabla^2_{yy} g(x,y)\right]^{-1} - \mathbb{E}[H_{hia}] \| &\leq \frac{1}{\mu_g} (1 - \mu_g \eta)^Q \\ \mathbb{E}[\| \left[\nabla^2_{yy} g(x,y)\right]^{-1} - H_{hia} \|] &\leq \frac{2}{\mu_g} \end{aligned}$$

where the expectation is taken with respect to p and the random indices $i_1, \dots, i_p$.

Proposition 3.3 shows that the bias of H_{hia} decreases linearly with Q. The estimate's variance is bounded, but the analysis does not take into account any batch size, preventing a fine control of this variance.

Ji et al. (2021) proposes with the algorithm stocBiO a modification of Equation (3.14) which involves matrix-vector multiplications rather than matrix-matrix multiplications and that aims at estimating $\left[\nabla^2_{yy} g(x,y)\right]^{-1} \nabla f(x,y)$. Let $\mathcal{B}_1, \dots, \mathcal{B}_Q$ be Q batches of samples drawn from $\mathbb{P}_g$ and $\mathcal{B}_f$ a batch of samples drawn from $\mathbb{P}_f$. The authors propose the following estimator

$$v_{\text{shia}} = \eta \sum_{q=-1}^{Q-1} \prod_{s=Q-q}^{Q} [I - \eta \nabla^2_{yy} g_{\mathcal{B}_s}(x,y)] \nabla_y f_{\mathcal{B}_f}(x,y) \tag{3.15}$$

where $\eta \in \left(0, \frac{1}{L_{g,1}}\right)$ and $Q \in \mathbb{N}$. The estimator v_{shia} comes with the following guarantees:

Proposition 3.4 (Ji et al. (2021, Proposition 3))**.** *Suppose that Assumptions 3.3 to 3.5 hold. Let $\eta \leq \frac{1}{L_{g,1}}$ and choose the batch size $|\mathcal{B}_{Q+1-q}| = BQ(1-\eta\mu_g)^{q-1}$ for $q \in [Q]$, where $B \geq \frac{1}{Q(1-\eta\mu_g)^{Q-1}}$. Then it holds*

$$\begin{aligned} \|\mathbb{E}[v_{shia}] - \hat{v}(x;y)\|^2 &\leq \frac{1}{\mu_g} (1 - \eta\mu_g)^{Q+1} L_{f,0} \\ \mathbb{E}[\|v_{shia} - \hat{v}(x;y)\|^2] &\leq \frac{4\eta^2}{\mu_g^2 L_{g,1} L_{f,0}^2} \frac{1}{B} + \frac{4(1-\eta\mu_g)^{2Q+2} L_{f,0}^2}{\mu_g^2} + \frac{2L_{f,0}^2}{\mu_g^2 |\mathcal{B}_f|} \quad . \end{aligned}$$

On the one hand, as for Proposition 3.3, Proposition 3.4 shows a linear decrease of the bias with respect to Q. On the other hand, the variance of v_{shia} is upper bounded by a term that decreases linearly with Q, a term that decreases sublinearly with B, and another that decreases sublinearly with the batch size $|\mathcal{B}_f|$. Consequently, to determine the complexity of stocBiO, the authors assumed batch sizes in $\mathcal{O}(\epsilon^{-1})$ to effectively mitigate the variance terms in their final result.

It is important to note that these Neumann approximation-based estimators are inherently cold-started, meaning we cannot reuse the estimator from the previous outer iteration. Consequently, algorithms employing Neumann approximations must set $Q = \Theta(\log(\epsilon^{-1}))$ to reach an ϵ-stationary solution (see Definition 3.1). In contrast, the following approach, which is based on the optimization formulation of the inverse Hessian-vector product (iHVP) computation, does not have this limitation.

Stochastic Gradient Descent. Since the g is assumed to be μ_g-strongly convex, for fixed $x \in \mathbb{R}^{d_x}$ and $y \in \mathbb{R}^{d_y}$, the Hessian matrix $\nabla^2_{yy} g(x,y)$ is symmetric positive definite. As a consequence, the iHVP $\hat{v}(x;y)$ can be cast as the minimization of a quadratic form

$$v \mapsto \frac{1}{2} v^\top \nabla^2_{yy} g(x,y) v + v^\top \nabla_y f(x,y) \ .$$

This enables the use of classical optimization algorithms to solve the linear system. In particular, Arbel and Mairal (2022a) use this formulation to warm-start the linear system solver, removing by the way the additional factor $\log(\epsilon^{-1})$ usually present in the complexity of solvers that use Neumann iterations (Ji et al., 2021; Chen et al., 2021). By doing so, Arbel and Mairal (2022a) manage to get a complexity in $\mathcal{O}(\epsilon^2)$, matching, by the way, the complexity of the stochastic gradient descent for non-convex single-level problems. This technique was also implemented in a one-loop fashion by Li et al. (2022) who propose to perform only one stochastic gradient step in the linear system.

Grazzi et al. (2023) also propose performing stochastic gradient steps to solve the linear system, initializing the solver from scratch at each iteration. Consequently, their analysis assume a number a linear system resolution steps in $\Theta(t)$ where t is the current outer iteration.

We have seen so far that stochastic AID-based algorithms are the result of mixing some ingredients: the choice of the inner solver, the choice of the linear system solver, the use or not of variance reduction techniques, the use of warm-start or not... In Table 3.1, we provide a comprehensive comparison of stochastic AID-based algorithms.

Method	iHVP	Inner loop	LR inner	LR outer	BS	WS Inner	WS iHVP	Number steps inner	Number steps iHVP	Complexity
BSA (Ghadimi and Wang, 2018)	HIA	SGD	$\mathcal{O}(k^{-1})$	$\mathcal{O}(T^{-\frac{1}{2}})$	$\Theta(1)$	✗	✗	$\Theta(\sqrt[4]{t})$	$\Theta(\log(t))$	$\mathcal{O}(\epsilon^{-3})$
stocBiO (Ji et al., 2021)	SHIA	SGD	1	1	$\Theta(\epsilon^{-1})$	✓	✗	$\Theta(1)$	$\Theta(\log(T))$	$\tilde{\mathcal{O}}(\epsilon^{-2})$
VRBO (Yang et al., 2021)	SHIA	SPIDER	1	1	$\Theta(1)$	✓	✗	$\Theta(1)$	$\Theta(\log(T))$	$\tilde{\mathcal{O}}(\epsilon^{-\frac{3}{2}})$
AmIGO (Arbel and Mairal, 2022a)	SGD	SGD	1	1	$\Theta(\epsilon^{-1})$	✓	✓	$\Theta(1)$	$\Theta(1)$	$\mathcal{O}(\epsilon^{-2})$
ALSET (Chen et al., 2021)	HIA	SGD	$\mathcal{O}(T^{-\frac{1}{2}})$	$\mathcal{O}(T^{-\frac{1}{2}})$	$\Theta(1)$	✓	✗	$\Theta(1)$	$\Theta(\log(T))$	$\tilde{\mathcal{O}}(\epsilon^{-2})$
BGSM (Grazzi et al., 2023)	SGD	SGD	$\mathcal{O}(T^{-\frac{1}{2}})$	$\mathcal{O}(T^{-\frac{1}{2}})$	$\Theta(1)$	✗	✗	$\Theta(t)$	$\mathcal{O}(t)$	$\tilde{\mathcal{O}}(\epsilon^{-2})$
TTSA (Hong et al., 2023)	HIA	SGD	$\mathcal{O}(T^{-\frac{2}{5}})$	$\mathcal{O}(T^{-\frac{3}{5}})$	$\Theta(1)$	✓	✗	1	$\Theta(\log(t))$	$\tilde{\mathcal{O}}(\epsilon^{-\frac{5}{2}})$
SMB (Guo et al., 2021a)	HIA	SGD with momentum	1	1	$\Theta(1)$	✓	✗	1	$\Theta(\log(T))$	$\tilde{\mathcal{O}}(\epsilon^{-4})$
MRBO (Yang et al., 2021)	SHIA	STORM	$\mathcal{O}(t^{-\frac{1}{3}})$	$\mathcal{O}(t^{-\frac{1}{3}})$	$\Theta(1)$	✓	✗	1	$\Theta(\log(T))$	$\tilde{\mathcal{O}}(\epsilon^{-\frac{3}{2}})$
STABLE (Chen et al., 2022a)	Direct	SGD with outer correction	$\mathcal{O}(T^{-\frac{1}{2}})$	$\mathcal{O}(T^{-\frac{1}{2}})$	$\Theta(1)$	✓	✗	1	NA	$\mathcal{O}(\epsilon^{-2})$
SUSTAIN (Khanduri et al., 2021)	HIA	STORM	$\mathcal{O}(t^{-1/3})$	$\Theta(1)$	$\mathcal{O}(t^{-\frac{1}{3}})$	✓	✗	1	$\Theta(\log(T))$	$\mathcal{O}(\epsilon^{-\frac{3}{2}})$
SVRB (Guo et al., 2021b)	Direct + momentum	SGD with momentum	$\mathcal{O}(t^{-\frac{1}{3}})$	$\mathcal{O}(t^{-\frac{1}{3}})$	$\Theta(1)$	✓	✗	1	NA	$\tilde{\mathcal{O}}(\epsilon^{-3})$
FSLA (Li et al., 2022)	SGD	SGD	$\mathcal{O}(t^{-\frac{1}{2}})$	$\mathcal{O}(T^{-\frac{1}{2}})$	$\Theta(1)$	✓	✓	1	1	$\mathcal{O}(\epsilon^{-2})$

Table 3.1: Comparison of the stochastic bilevel optimization solvers in the literature. The complexity is the number of oracle calls necessary to attain an ϵ-accurate stationary point (see Definition 3.1). iHVP stands for inverse Hessian-vector product. LR stands for learning rate. BS stands for batch size. Learning rate equals 1 means constant learning rate, independent from the horizon. Learning rate expressed in T means a fixed learning rate that depends on the horizon of the outer loop. Learning rate in t means decreasing learning rate with respect to the outer iteration t. Learning rate in k means decreasing learning rate with respect to the inner iteration k.

Aside from the AID-based methods, other algorithms leveraging automatic differentiation have emerged in the literature. Although this thesis is focused on AID-based methods, we briefly present this line of work in the next section for completeness.

3.3 Iterative Differentiation

In the hypergradient given Equation (3.1), the Jacobian of y^* is computed by leveraging the Implicit Function Theorem. This method requires solving approximately a potentially large linear system that involves the Hessian matrix of the inner function. A second line of work considered another class of method to approximate the Jacobian of y^*: the Iterative Differentiation (ITD).

In the bilevel optimization context, ITD was first proposed by Domke (2012) for energy-based models. ITD-based algorithms consist of differentiating the different steps of an iterative solver that solves the inner problem. Domke (2012) proposes to apply the backpropagation algorithm to the gradient descent and or the heavy ball steps to approximate the Jacobian of y^*. This idea of differentiating through optimization steps, also known as "unrolling", is also popular in the inverse problem literature (Bertocchi et al., 2020; Brauer et al., 2023; Huang et al., 2023; Bonettini et al., 2024).

One drawback of this method is the memory cost of the backpropagation that grows linearly with the number of steps. One way to circumvent this is to use gradient checkpointing (Hascoet and Araya-Polo, 2006). It consists in storing only a subset of the iterates and recomputing the other iterates on-the-fly. Another technique was proposed by Maclaurin et al. (2015) where the authors make the observation that the algorithm SGD with momentum is reversible. This means that starting from the output of the algorithm, one can rebuild the optimization path. This allows to apply the backpropagation algorithm without storing the intermediate iterates. Instead, the intermediate steps are recomputed on the fly. However, the recomputation of the iterates adds an overhead to the algorithm's iterations, making it slower. Shaban et al. (2019) propose to backpropagate only in the K last steps of the inner algorithm. The authors show that the approximation error of the hypergradient decreases exponentially with K. Bolte et al. (2023) show the efficiency of one-step iterative differentiation (that is, differentiating only the last optimization step of the inner solver) for superlinear optimization methods.

Several authors also studied the consistency of the approximation of the Jacobian of y^* by the sequences of Jacobian $(\mathrm{d}y^k)_{k\geq 0}$ where y^k is the k-th iterate of an iterative that approximates y^*. In particular, Gilbert (1992) shows that if the sequence $(y^k)_{k\geq 0}$ is generated by iterating a contracting operator, then the sequence of the Jacobians $(\mathrm{d}y^k)_{k\geq 0}$ converges towards $\mathrm{d}y^*$. Mehmood and Ochs (2019) show the convergence of the Jacobians for the specific cases of gradient descent and heavy ball algorithms. Iutzeler et al. (2024) studies the behavior of the Jacobian in the case of stochastic gradient descent by showing the convergence of the Jacobian when using vanishing step sizes.

Grazzi et al. (2020) derive convergence rates for the estimation of the hypergradient in the AID and the ITD cases, depending on the number of iterations for the inner algorithm and the algorithm to solve the linear system (in the case of AID). Ablin et al. (2020) derive convergence rate in the particular case of min-min problems (that is $f = g$).

Scieur et al. (2022) study on quadratic objectives the effect of the choice of the step size for the convergence of the iterates of an optimization algorithm and their Jacobians. They show that large step sizes lead to fast convergence of the iterates but a long burn-in phase for the Jacobians. On the other hand, taking small step sizes leads to a slower convergence of the iterates but reduces the burn-in phase.

Arbel and Mairal (2022b) provide a theoretical framework of iterative differentiation when the inner problem has several solutions, which can occur in non-convex settings. This is done by the introduction of the concept of selection map, denoted as $\phi : \mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$, which selects one solution to the inner problem. One specific instance they discuss is when $\phi(x, y)$ is defined as the limit of the gradient flow applied to $g(x, \cdot)$, starting from the point y.

When the inner function is nonsmooth, the use of iterative differentiation has been proposed by Ochs et al. (2015, 2016) and the convergence properties of the Jacobians are studied by Bolte et al. (2022). However, Malézieux et al. (2022) show that the sequence of Jacobians is not always well-behaved, in particular when the support of the solution is badly estimated.

3.4 Penalty methods

Recently, another line of work studied the reformulation of the bilevel problems as single-level constrained optimization problems. This leads to the following formulation

$$\min_{(x,y)\in\mathbb{R}^{d_x}\times\mathbb{R}^{d_y}} f(x,y) \quad \text{s.t.} \quad g(x,y) \leq g^*(x) \ . \tag{3.16}$$

where $g^*(x) = \min_{y\in\mathbb{R}^{d_y}} g(x,y)$. From this formulation, Kwon et al. (2023a) consider the lagrangian

$$\mathcal{L}_\lambda(x,y) = f(x,y) + \lambda(g(x,y) - g^*(x))$$

which consists in penalizing the outer function by the violation of $g(x,y)$ being far from $g^*(x)$. They propose to apply (stochastic) gradient descent to this lagrangian while increasing the Lagrange multiplier λ. The particularity of this method is that it does not require second-order information from the inner function g since, by Danskin's theorem (Danskin, 1967, Theorem 1), the gradient of g^* is given by

$$\nabla g^*(x) = \nabla_x g(x, y^*(x)) \ .$$

The analysis of this kind of method is refined in subsequent papers (Kwon et al., 2023b; Chen et al., 2023a). Moreover, Kwon et al. (2024) provide a lower complexity bound for bilevel problem resolution where the algorithm is assumed to access only first-order information of the inner and outer functions and the solution of the inner problem y^* is assumed to be accessed up to a certain precision.

Liu et al. (2023) propose a different approach where the constrained in Problem (3.16) is replaced by a first-order condition on the inner problem instead of the inner suboptimality gap

$$\min_{(x,y)\in\mathbb{R}^{d_x}\times\mathbb{R}^{d_y}} f(x,y) \quad \text{s.t.} \quad \nabla_y g(x,y) = 0 \ . \tag{3.17}$$

They also propose to solve Problem (3.17) by seeking a KKT point of the associated Lagrangian.

3.5 Benchmarking bilevel optimization algorithms

3.5.1 Presentation

The previous presentation emphasizes the theoretical guarantees of performance for bilevel optimization algorithms. However, it is also important to assess their practical performance to understand the benefits and limitations of each method, and to reinforce theoretical claims (Sculley et al., 2018). Theoretical guarantees are often insufficient to predict practical performance for the following reasons:

- **Worst-case analysis.** The complexities in the literature are worst-case complexities, providing an upper bound on the computation needed to reach an approximate solution. However, the worst-case scenario can differ significantly from most practical applications.
- **Theoretical guarantees ignore the downstream task.** The guarantees we get quantify the efficiency of an algorithm in solving a bilevel optimization problem. For bilevel problems, this often means quantifying how effectively an algorithm approaches a stationary point. However, in practice, the ultimate goal of using bilevel algorithms is not merely to find a stationary point but to perform a downstream task.

- **Assumptions.** Some assumptions used to prove theoretical results may not hold in practice. For example, the assumption of a constant $\mu_g > 0$ such that the inner function $g(x, \cdot)$ is strongly convex for all $x \in \mathbb{R}^{d_x}$ can be violated. This occurs, for instance, in the case of Ridge regression loss in the overparametrized regime, where the inner variable is the model's parameters and the outer variable is the regularization parameter.

- **Hardware specifications.** Theoretical results often ignore the characteristics of the hardware used to run the algorithms. Depending on the capacity to parallelize computations or manage memory, the performance of the algorithms can vary significantly.

Bilevel optimization papers provide empirical results on classical tasks such as hyperparameter selection or data cleaning. However, these results often lack reproducibility. The code is frequently unavailable, or if it is available, its documentation is incomplete. Indeed, ingredients necessary for a proper reimplementation may be missing, such as important algorithmic details, or hyperparameter choices (Pineau et al., 2019). Additionally, the same algorithm can exhibit varying performances across different papers for the same task, reducing trust in the empirical results. Moreover, it is important for the practitioners to have a clear understanding of which algorithm is best suited for specific tasks (Bartz-Beielstein et al., 2020). To address these issues, it is crucial to develop and maintain an up-to-date benchmark for bilevel optimization algorithms.

We have initiated this effort with the bilevel benchmark we have built over the last three years. This benchmark is constructed using the `Python` package `Benchopt` (Moreau et al., 2022). `Benchopt` is a tool that enables to easily build and maintain benchmarks of optimization algorithms. The benchmarks are built in an open, standardized, and collaborative way so that researchers can use it to benchmark their algorithm, without the burden of reimplementing baseline methods from the literature. The code of the benchmark is open source and available on GitHub.[1] It enjoys the classical `Benchopt` features such as parallel runs, caching, and automatical results archiving, making its execution easy.

3.5.2 Details on the benchmark

Solver. The benchmark contains fourteen solvers:

- **Four stochastic AID-based solvers without variance reduction.** AmIGO (Arbel and Mairal, 2022a), stocBiO (Ji et al., 2021), BSA (Ghadimi and Wang, 2018), TTSA (Hong et al., 2023);

- **Four stochastic AID-based solvers with variance reduction.** MRBO (Yang et al., 2021), VRBO (Yang et al., 2021), SUSTAIN (Khanduri et al., 2021), FSLA (Li et al., 2022);

- **Two penalty-based solvers (one stochastic and one deterministic).** BOME (Ye et al., 2022), F2SA (Kwon et al., 2023a);

- **One Hessian-free solver.** PZOBO (Sow et al., 2022);

- **Two full-batch solvers based on Jaxopt (Blondel et al., 2021).** Jaxopt-GD (AID-based), Jaxopt-ITD (ITD-based);

- **A zero-order solver.** `Optuna` (Akiba et al., 2019).

Apart from the Jaxopt and Optuna solvers, all the other were implemented from scratch in `Jax` (Bradbury et al., 2018). This enables the use of the automatic differentiation tools to compute the different oracles, to leverage the just-in-time compilation of `Jax`, and the automatic vectorization of functions.

[1] https://github.com/benchopt/benchmark_bilevel

Tasks. The benchmark contains four different bilevel problems:

- **Quadratic functions.** This toy problem aims at having a problem setting where one can control the different constants of the problem. We consider a setting where the inner and the outer functions are sums of quadratics.

$$f(x,y) = \frac{1}{m}\sum_{j=1}^{m} f_j(x,y), \quad g(x,y) = \frac{1}{n}\sum_{i=1}^{n} g_i(x,y) \ .$$

The functions f_j and g_i are defined as

$$f_j(x,y) = \frac{1}{2}y^\top A_y^{f_j} y + \frac{1}{2}x^\top A_x^{f_j} + xB^{f_j}y + (d_y^{f_j})^\top y + (d_x^{f_j})^\top x$$
$$g_i(x,y) = \frac{1}{2}y^\top A_y^{g_i} y + \frac{1}{2}x^\top A_x^{g_i} + xB^{g_i}y + (d_y^{g_i})^\top y + (d_x^{g_i})^\top x$$

with $A_y^{f_j}, A_y^{g_i} \in \mathbb{R}^{p\times p}$, $A_x^{f_j}, A_x^{g_i} \in \mathbb{R}^{d\times d}$, $B^{f_j}, B^{g_i} \in \mathbb{R}^{d\times p}$, $d_y^{f_j}, d_y^{g_i} \in \mathbb{R}^p$ and $d_x^{f_j}, d_x^{g_i} \in \mathbb{R}^d$. The vectors $d_x^{f_j}$, $d_x^{g_i}$ are drawn randomly according to a normal distribution $\mathcal{N}(0, I_d)$. The vectors $d_y^{f_j}$, $d_y^{g_i}$ are drawn randomly according to a normal distribution $\mathcal{N}(0, I_p)$. For the Hessian matrices with respect to y, we generate $A_y^{g_i}$ so that $\frac{1}{n}\sum_{i=1}^n A_y^{g_i} = A$ for a symetric positive definite matrix A with a given spectrum. To do so, we generate $x_i \sim \mathcal{N}(0, I_p)$ and set $A_y^{g_i} = \sqrt{A}x_i(\sqrt{A}x_i)^\top$. We proceed similarly for $A_y^{f_j}$, $A_y^{g_i}$, $A_x^{f_j}$. For B^{g_i}, we want $\frac{1}{n}\sum_{i=1}^n B^{g_i} = B$ for a prescribed matrix $B \in \mathbb{R}^{d\times p}$ such that with a prescribed spectral norm $\|B\|$. Let $B = U\Sigma V^\top$ the singular values decomposition of B. To get B^{g_i}, we generate $x_i \sim \mathcal{N}(0, I_p)$ and set $B^{g_i} = (V\Sigma x_i)(Ux_i)^\top$. We proceed similarly for B^{f_j}.

- **Hyperparameter selection with IJCNN1.** The IJCNN1 dataset[2] is a dataset for binary classification. It has $n = 49,990$ training samples, $m = 91,701$ validation samples, and $p = 22$ features. For this dataset, we want to select a regularization parameter for an ℓ^2-regularized logistic regression problem where we have one hyperparameter per feature. This leads to the following outer and inner functions

$$f(\lambda,\theta) = \frac{1}{m}\sum_{j=1}^{m} \log(1+\exp(-y_j^{\text{val}}\langle\theta, x_j^{\text{val}}\rangle))$$
$$g(\lambda,\theta) = \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp(-y_i^{\text{train}}\langle\theta, x_i^{\text{train}}\rangle)) + \frac{1}{2}\sum_{k=1}^{p} \mathrm{e}^{\lambda_k}\theta_k^2$$

where the inner variable $\theta \in \mathbb{R}^p$ denotes the model's parameter and the outer variable $\lambda \in \mathbb{R}^p$ denotes the hyperparameter.

- **Hyperparameter selection with COVTYPE.** The COVTYPE dataset [3] contains $581,012$ samples with $p = 54$ features distributed into $C = 7$ classes. We fit a multiclass ℓ^2-regularized logistic regression with one hyperparameter per feature. The inner and outer functions are defined as

$$f(\lambda,\theta) = \frac{1}{m}\sum_{j=1}^{m} \ell(\theta x_j^{\text{val}}, y_j^{\text{val}})$$
$$g(\lambda,\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(\theta x_i^{\text{val}}, y_i) + \sum_{c=1}^{C} \mathrm{e}^{\lambda_c}\sum_{k=1}^{p}\theta_{k,c}^2$$

where $\theta \in \mathbb{R}^{p\times C}$ is the model's parameter and $\lambda \in \mathbb{R}^C$ is the hyperparameter. We used $n = 371,847$ training samples, $m = 92,962$ validation samples, and $n_{\text{test}} = 116,203$ test samples.

[2] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html
[3] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_covtype.html

- **Data hypercleaning with MNIST.** The datacleaning task presented in Section 1.3 is performed with the MNIST dataset [4]. The dataset consists of $n = 60000$ training samples and $m = 10000$ validation samples au size 28×28. The inner and outer functions are defined as

$$f(w, \theta) = \frac{1}{m} \sum_{j=1}^{m} \ell(\theta x_j^{\text{val}}, y_j^{\text{val}})$$

$$g(w, \theta) = \frac{1}{n} \sum_{i=1}^{n} \sigma(w_i) \ell(\theta x_i^{\text{val}}, y_i) + C_r \|\theta\|_2^2$$

  where ℓ is the cross-entropy loss and $C_r > 0$ is a regularization parameter ($C_r = 0.2$ in our case). For this task, a proportion $p \in \{0.5, 0.7, 0.9\}$ of the labels of the training samples are corrupted, as illustrated in figure 1.6. The labels of the validation and test sets remain unchanged.

Selection of the algorithms' hyperparameters. The algorithms we consider in our benchmark come with their own hyperparameters. Depending on the methods, those hyperparameters can include: the inner or outer step size, the number of inner steps, the momentum parameter... To select these hyperparameters, we perform an extensive grid search. For each solver and task, we keep the hyperparameters that perform the best over the 100 first iterations. This choice of looking at the first iteration for curve selection relies on a tradeoff we want between the final performance of the algorithm and its execution speed. The measure of performance can be the validation accuracy (for the hyperparameter selection and data cleaning tasks) or the value of the objective function (for the quadratic task). Also, for stochastic solvers, we perform 10 runs with different random seeds. Among these 10 runs, it is possible that some of them crash because of numerical instability. In order to guarantee the robustness of the hyperparameter selected, we select the hyperparameters among those that have not led to a crash in any of the 10 runs.

Examples of results. In figure 3.4, we provide an example of a performance curve we can get with the benchmark on quadratics. For this setting, we display the squared norm of the value function Φ. We consider a problem where the inner dimension is 100, and the outer dimension is 10. Moreover, the inner sum contains $n = 8192$ functions and the outer sum contains $m = 1024$ functions. In figure 3.4, one observes that the slower solver is `Optuna`. This result is expected because of the dimensionality of the problem. Additionally, we notice that Hessian-free solvers are slower compared to the AID-based solvers. Among the stochastic solvers, AMIGO achieves the best final performance, likely because it is fully warm-started; both the inner and linear system solvers are warm-started. Conversely, the deterministic solver Jaxopt GD, while slower, reaches a better final value as it avoids getting stuck in a plateau.

In figure 3.5, we display the performance curves of various solvers on the hyperparameter selection task using the COVTYPE dataset. It is observed that the deterministic and zero-order solvers are the slowest compared to the stochastic ones. Moreover, they achieve a higher final test error. The better performance of the stochastic solvers could be attributed to the non-convexity of the problem and the stochastic nature of the solver, which may help reach a better solution.

[4] http://yann.lecun.com/exdb/mnist/

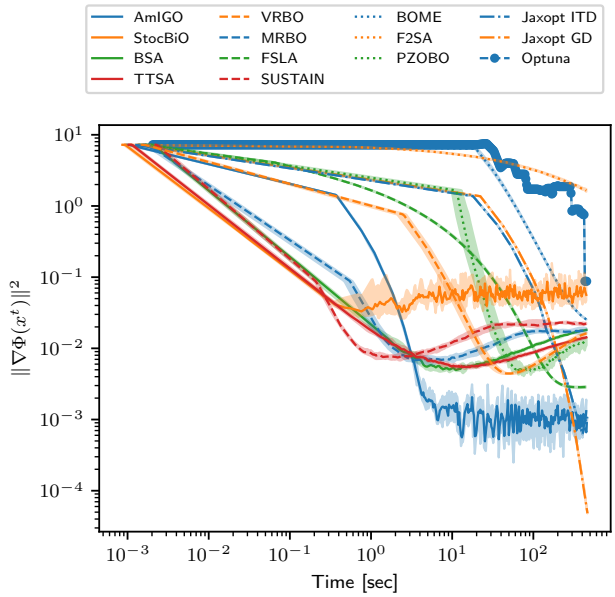


Figure 3.4: Median of the squared norm of the gradient of the value function Φ over 10 runs for the different solvers on the benchmark on quadratics in function of wall-clock time. The shaded area corresponds to the performances between the 20% and the 80% percentiles. The inner variable has a dimension of 100, and the outer variable has a dimension of 10. The solvers in plain lines are the stochastic solvers, the solvers in dashed lines are the variance-reduced solvers, the solvers in dotted lines are the Hessian-free solvers, the solvers in dash-dotted lines are the deterministic solvers and the solver with big dot markers is the zero-order solver.

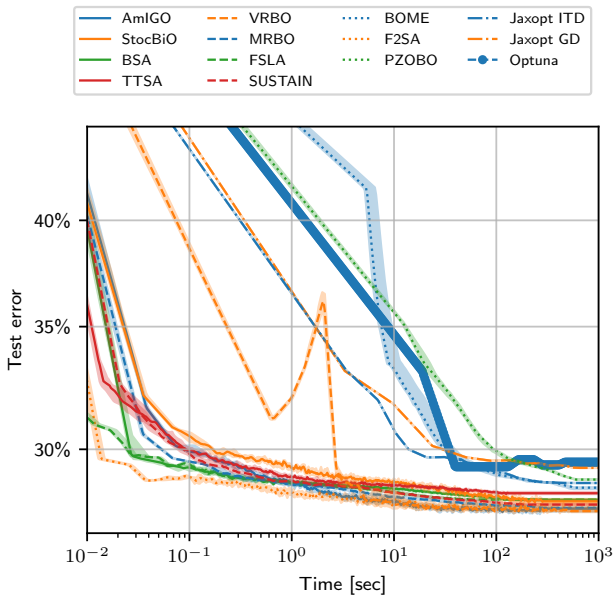


Figure 3.5: Median of the test error over 10 runs for the different solvers on the benchmark on hyperparameter selection with the COVTYPE dataset in function of wall-clock time. The shaded area corresponds to the performances between the 20% and the 80% percentiles. The solvers in plain lines are the stochastic solvers, the solvers in dashed lines are the variance-reduced solvers, the solvers in dotted lines are the Hessian-free solvers, the solvers in dash-dotted lines are the deterministic solvers and the solver with big dot markers is the zero-order solver.

Part II

Contributions

CHAPTER 4

A FRAMEWORK FOR BILEVEL OPTIMIZATION THAT ENABLES STOCHASTIC AND GLOBAL VARIANCE REDUCTION ALGORITHMS

Sommaire

This section presents the work selected for an oral at NeurIPS 2022:

M. Dagréou, P. Ablin, S. Vaiter, and T. Moreau. A framework for bilevel optimization that enables stochastic and global variance reduction algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Only minor modifications have been made from the original paper: adaptation of the notation, shorter introduction, relocation of some proofs in the main text, or addition of proof sketch for results with proof in the appendix (Chapter A) relocation of experimental details in the main text.

4.1 Introduction

In this chapter, we focus on bilevel optimization problem where the inner problem has a single solution, that is

$$\min_{x\in\mathbb{R}^{d_x}} \Phi(x) = f(x, y^*(x)), \quad \text{such that } y^*(x) = \arg\min_{y\in\mathbb{R}^{d_y}} g(x,y) \ , \tag{4.1}$$

In the introduction, we have seen that under appropriate hypotheses, the function Φ is differentiable and that its gradient is given for any $x \in \mathbb{R}^{d_x}$ by

$$\nabla\Phi(x) = \nabla_x f(x, y^*(x)) + \nabla^2_{xy} g(x, y^*(x)) v^*(x) \ , \tag{4.2}$$

where $v^*(x) \in \mathbb{R}^{d_y}$ is the solution of a linear system

$$v^*(x) = -\left[\nabla^2_{yy} g(x, y^*(x))\right]^{-1} \nabla_y f(x, y^*(x)) \ . \tag{4.3}$$

In the light of (4.2) and (4.3), it turns out that the derivation of the gradient of Φ at each iteration is cumbersome because it involves two subproblems: the resolution of the inner problem to find an approximation of $y^*(x)$ and the resolution of a linear system to find an approximation of $v^*(x)$. It makes the practical implementation of first-order methods like gradient descent for (4.1) challenging.

As is the case in many machine learning problems, we suppose in this chapter that f and g are empirical means:

$$f(x, y) = \frac{1}{m}\sum_{j=1}^{m} f_j(x, y), \quad g(x, y) = \frac{1}{n}\sum_{i=1}^{n} g_i(x, y) \ .$$

This structure suggests the use of stochastic methods to solve (4.1). For single-level problems (that is, classical optimization problems where one function should be minimized), using Stochastic Gradient Descent (SGD; (Robbins and Monro, 1951; Bottou, 2010)) and variants is natural because individual gradients are straightforward unbiased estimators of the gradient. In the bilevel framework, we want to develop algorithms that make progress on problem (4.1) by using only a few functions f_j and g_i at a time. However, since $\nabla\Phi$ involves the inverse of the Hessian of g, building such stochastic algorithms is quite challenging, one of the difficulties being that there is no straightforward unbiased estimator of $\nabla\Phi$. Still, in settings where m or n are large, where computing even a single evaluation of f or g is extremely expensive, stochastic methods are the only scalable algorithms.

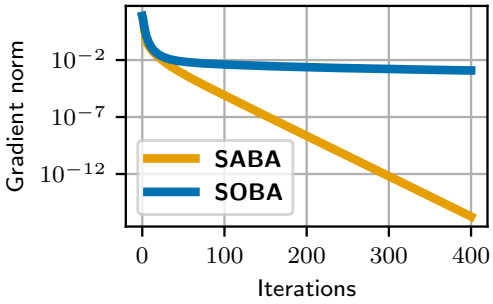


Figure 4.1: Convergence curves of the two proposed methods on a toy problem. SABA is a stochastic method that achieves fast convergence on the value function.

Variance reduction (Johnson and Zhang, 2013; Defazio et al., 2014; Schmidt et al., 2017; Fang et al., 2018; Cutkosky and Orabona, 2019) is a popular technique to obtain fast stochastic algorithms. In a single-level setting, these methods build an approximation of the gradient of the objective function using only stochastic gradients. Contrary to SGD, the variance of the approximation goes to 0 as the algorithm progresses, allowing for faster convergence. For instance, the SAGA method (Defazio et al., 2014) achieves linear convergence if the objective function satisfies a Polyak-Łojasciewicz inequality, and $\mathcal{O}(\frac{1}{T})$ convergence rate on smooth non-convex functions (Reddi et al., 2016). The extension of these methods to bilevel optimization is a natural idea to develop faster algorithms. As we have seen in Chapter 3, the landscape of variance-reduced algorithms is restricted. Indeed, only the STORM (Cutkosky and Orabona, 2019) and the SPIDER (Fang et al., 2018) algorithms have been extended to the bilevel setting by Yang et al. (2021), Khanduri et al. (2021) and Li et al. (2022) for STORM and by Yang et al. (2021) for SPIDER. This can be explained by the fact that this idea is hard to implement because it is hard to derive unbiased estimators of $\nabla\Phi$, let alone variance reduction ones.

Contributions of Chapter 4. In this chapter, we introduce a novel framework for bilevel optimization in Section 4.2, where the inner variable, the solution of the linear system (4.3) and the outer variable evolve jointly. The evolution directions are written as sums of derivatives of f_j and g_i, which allows us to derive simple unbiased stochastic estimators. In this framework, we propose SOBA, an extension of SGD (subsection 4.2.1), and SABA (subsection 4.2.2), an extension of the variance reduction algorithm SAGA (Defazio et al., 2014). In Section 4.3, we analyze the convergence of our methods.

SOBA is shown to achieve $\inf_{t\leq T}\mathbb{E}[\|\nabla\Phi(x^t)\|^2] = \mathcal{O}(\log(T)T^{-\frac{1}{2}})$ with decreasing step sizes. We prove that SABA with fixed step sizes achieves $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\Phi(x^t)\|^2] = \mathcal{O}(\frac{1}{T})$. SABA is therefore, to the best of our knowledge, the first stochastic bilevel algorithm that matches the convergence rate of gradient descent on Φ. We also prove that SABA achieves linear convergence under the assumption that Φ satisfies a Polyak-Łojasciewicz inequality. To the best of our knowledge, SABA is also the first stochastic bilevel algorithm to feature such a property. Importantly, these rates match the rates of the single-level counterparts of each algorithm in a non-convex setting (SGD for SOBA and SAGA for SABA). Finally, in Section 4.4, we provide an extensive benchmark of many stochastic bilevel methods on hyperparameters selection and data hypercleaning, and illustrate the usefulness of our approach.

4.2 Proposed framework

Algorithm 5 General framework

Input: initializations $y_0 \in \mathbb{R}^{d_y}$, $x_0 \in \mathbb{R}^{d_x}$, $v_0 \in \mathbb{R}^{d_y}$, number of iterations T, step size sequences $(\rho^t)_{t<T}$ and $(\gamma^t)_{t<T}$.
for $t = 0, \dots, T-1$ **do**
  Update y: $y^{t+1} = y^t - \rho^t D_y^t$,
  Update v: $v^{t+1} = v^t - \rho^t D_v^t$,
  Update x: $x^{t+1} = x^t - \gamma^t D_x^t$,
  where D_y^t, D_v^t and D_x^t are unbiased estimators of the directions $D_y(y^t, v^t, x^t)$, $D_v(y^t, v^t, v^t)$ and $D_x(y^t, v^t, x^t)$.
end for

In this section, we introduce our framework in which the solution of the inner problem, the solution of the linear system (4.3), and the outer variable all evolve at the same time, following directions that are written as a sum of derivatives of f_j and g_i. We define

$$D_y(y, v, x) = \nabla_y g(x, y) \ , \tag{4.4a}$$
$$D_v(y, v, x) = \nabla^2_{yy} g(x, y)v + \nabla_y f(x, y) \ , \tag{4.4b}$$
$$D_x(y, v, x) = \nabla^2_{xy} g(x, y)v + \nabla_x f(x, y) \ . \tag{4.4c}$$

These directions are motivated by the fact that we have $\nabla\Phi(x) = D_x(y^*(x), v^*(x), x)$, with $y^*(x)$ the minimizer of $g(x, \cdot)$ and $v^*(x)$ the solution of $\nabla^2_{yy} g(x, y^*(x))v = -\nabla_y f(x, y^*(x))$. When x is fixed, we approximate y^* by doing a gradient step on g, following the direction $-D_y(y, v, x)$. Finally, when y and x are fixed, we find v^* by following the direction $-D_v(y, v, x)$, which corresponds to a gradient descent on $v \mapsto \frac{1}{2}\langle\nabla^2_{yy} g(x, y)v, v\rangle + \langle\nabla_y f(x, y), v\rangle$. The rest of the chapter is devoted to the study of the global dynamics where the three variables y, v and x evolve at the same time, following stochastic approximations of D_y, D_v and D_x. The next proposition motivates the choice of these directions.

Proposition 4.1. *Assume that for all $x \in \mathbb{R}^{d_x}$, the inner function $g(x, \cdot)$ is strongly convex. If (y, v, x) is a zero of (D_y, D_v, D_x), then $y = y^*(x)$, $v = v^*(x)$ and $\nabla\Phi(x) = 0$.*

Proof. Let (y, v, x) a zero of (D_y, D_v, D_x). For D_y, this means that $\nabla_y g(x, y) = 0$. Since $g(x, \cdot)$ is strongly convex, y is the minimizer of $g(x, \cdot)$, *i.e.* $y = y^*(x)$. The fact that (y, v, x) is a zero of D_v implies that $\nabla^2_{yy} g(y, x)v = -\nabla_y f(x, y)$. Replacing y by its value, we get $v = -\left[\nabla^2_{yy} g(x, y^*(x))\right]^{-1}\nabla_y f(x, y^*(x))$ which is $v^*(x)$ by definition. Putting all together and using the expression of $\nabla\Phi(x)$ given by (4.2), we get

$$D_x(y, v, x) = \nabla_x f(x, y^*(x)) + \nabla^2_{xy} g(x, y^*(x))v^*(x) = \nabla\Phi(x) \ .$$

On the other hand, $D_x(y, v, x) = 0$ so $\nabla\Phi(x) = 0$. □

We also note that the computation of these directions does *not* require computing the Hessian matrices $\nabla^2_{yy} g(x,y)$ and $\nabla^2_{xy} g(x,y)$: we only need to compute their product with a vector, which can be computed at a cost similar to that of computing a gradient.

The framework we propose is summarized in Algorithm 5. It consists in following a joint update rule in (y,v,x) that follows directions D^t_y, D^t_v and D^t_x that are unbiased estimators of D_y, D_v, D_x. The first and most important remark is that whereas $\nabla\Phi$ cannot be written as a sum over samples, the directions D_y, D_v and D_x involve only simple sums, since their expressions are "linear" in f and g:

$$
\begin{aligned}
D_y(y,v,x) &= \frac{1}{n}\sum_{i=1}^{n} \nabla_y g_i(x,y) \ , \\
D_v(y,v,x) &= \frac{1}{n}\sum_{i=1}^{n} \nabla^2_{yy} g_i(x,y)v + \frac{1}{m}\sum_{j=1}^{m} \nabla_y f_j(x,y) \ , \\
D_x(y,v,x) &= \frac{1}{n}\sum_{i=1}^{n} \nabla^2_{xy} g_i(x,y)v + \frac{1}{m}\sum_{j=1}^{m} \nabla_x f_j(x,y) \ .
\end{aligned}
$$

It is, therefore, straightforward to derive unbiased estimators of these directions. Li et al. (2022) considered one particular case of our framework, where the outer direction is estimated by using the STORM variance reduction technique (see (Cutkosky and Orabona, 2019)). Taking a step back by proposing the framework summarized in Algorithm 5 opens the way to potential new algorithms that implement other techniques that exist in stochastic single-level optimization. In what follows, we study two of them.

4.2.1 First example: the SOBA algorithm

The simplest unbiased estimator is obtained by replacing each mean by one of its terms chosen uniformly at random, akin to what is done in classical single-level SGD. We call the resulting algorithm SOBA (StOchastic Bilevel Algorithm). To do so, we choose two independent random indices $i \in [n]$ and $j \in [m]$ uniformly and estimate each term coming from g using g_i and each term coming from f using f_j. This gives the unbiased **SOBA directions**

$$
\boxed{
\begin{aligned}
D^t_y &= \nabla_y g_i(x^t, y^t) \ , && \text{(4.5)a} \\
D^t_v &= \nabla^2_{yy} g_i(x^t, y^t) v^t + \nabla_y f_j(x^t, y^t) \ , && \text{(4.5)b} \\
D^t_x &= \nabla^2_{xy} g_i(x^t, y^t) v^t + \nabla_x f_j(x^t, y^t) \ . && \text{(4.5)c}
\end{aligned}
}
$$

This provides us with a first algorithm, SOBA, where we plug Equations (4.5)a, (4.5)b and (4.5)c in Algorithm 5. We defer its analysis to the next section. Importantly, we use different step sizes for the update in (y,v) and for the update in x. We use the same step size in y and in v since the inner problem and the linear system have similar conditioning, which is that of $\nabla^2_{yy} G(x^t, y^t)$. The need for a different step size for the outer and inner problems is clear: both problems can have different conditioning.

An important remark for SOBA is that all the stochastic directions used are computed at the same point y^t, v^t and x^t with the same indices (i,j). The update of y, v and x can thus be performed in parallel instead of sequentially, benefiting from hardware parallelism. Moreover, this enables the sharing of the computations between the different directions. This is the case in hyperparameters selection where $g_i(x,y) = \ell_i(\langle y, d_i\rangle) + \frac{x}{2}\|y\|^2$, with d_i a training sample, and ℓ_i that measures how good is the prediction $\langle y, d_i\rangle$. In this setting, we have $\nabla_y g_i(x,y) = \ell_i'(\langle y, d_i\rangle) d_i + xy$ and $\nabla^2_{yy} g_i(x,y)v = \ell_i''(\langle y, d_i\rangle)\langle v, d_i\rangle d_i$. The prediction $\langle y, d_i\rangle$ can thus be computed only once to obtain both quantities. For more complicated models, where automatic differentiation is used to compute the different derivatives and Jacobian-vector products, we can store the computational graph only once to compute at the same time $\nabla_y g_i(x,y), \nabla^2_{yy} g_i(x,y)v$ and $\nabla^2_{21} g_i(x,y)v$, requiring only one backward pass, thanks to the $\mathcal{R}$ technique (Pearlmutter, 1994). This is explained in detail in Section 2.3.

Like all single-loop bilevel algorithms, our method updates at the same time the inner and outer variable, avoiding unnecessary optimization of the inner problem when x is far from the optimum.

4.2.2 Global variance reduction with the SABA algorithm

In classical optimization, SGD fails to reach optimal rates because of the variance of the gradient estimator. Variance reduction algorithms aim at reducing this variance, in order to follow directions that are closer to the true gradient and to achieve superior practical and theoretical convergence.

In our framework, since the directions D_y, D_v, and D_x are all written as sums of derivatives of f_j and g_i, it is easy to adapt most classical variance reduction algorithms. We focus on the celebrated SAGA algorithm (Defazio et al., 2014). The extension we propose is called SABA (Stochastic Average Bilevel Algorithm). The general idea is to replace each sum in the directions D by a sum over a memory, updating only one term at each iteration. To help the exposition, we denote $z = (y, x, v)$ the vector of joint variables. Since we have sums over i and over j, we have two memories for each variable: w_i^t for $i \in [n]$ and $\tilde{w}_j^t$ for $j \in [m]$, which keep track of the previous values of the variable y.

At each iteration t, we draw two random independent indices $i \in [n]$ and $j \in [m]$ uniformly and update the memories. To do so, we put $w_i^{t+1} = z^t$ and $w_{i'}^{t+1} = w_{i'}^t$ for $i' \neq i$, and $\tilde{w}_j^{t+1} = z^t$ and $\tilde{w}_{j'}^{t+1} = \tilde{w}_{j'}^t$ for $j' \neq j$. Each sum in the directions D is then approximated using SAGA-like rules: given n functions $\phi_{i'}$ for $i' \in [n]$, we define

$$S[\phi, w]_i^t = \phi_i(w_i^{t+1}) - \phi_i(w_i^t) + \frac{1}{n}\sum_{i'=1}^{n} \phi_{i'}(w_{i'}^t) \ .$$

This is an unbiased estimator of the average of the ϕ's since

$$\mathbb{E}_i\Big[S[\phi, w]_i^t\Big] = \frac{1}{n}\sum_{i'=1}^{n} \phi_{i'}(z^t) - \frac{1}{n}\sum_{i'=1}^{n} \phi_{i'}(w_{i'}^t) + \frac{1}{n}\sum_{i'=1}^{n} \phi_{i'}(w_{i'}^t) = \frac{1}{n}\sum_{i'=1}^{n} \phi_{i'}(z^t) \ .$$

With a slight abuse of notation, we call $\nabla^2_{yy} gv$ the sequence of functions $(y \mapsto \nabla^2_{yy} g_i(x, y)v)_{i\in[n]}$ and $\nabla^2_{xy} gv$ the sequence of functions $(y \mapsto \nabla^2_{xy} g_i(x, y)v)_{i\in[n]}$. We define the **SABA directions** as

$$\boxed{\begin{aligned} D_y^t &= S[\nabla_y g, w]_i^t \ , && \text{(4.6)a}\\ D_v^t &= S[\nabla^2_{yy} gv, w]_i^t + S[\nabla_y f, \tilde{w}]_j^t \ , && \text{(4.6)b}\\ D_x^t &= S[\nabla^2_{xy} gv, w]_i^t + S[\nabla_x f, \tilde{w}]_j^t \ . && \text{(4.6)c} \end{aligned}}$$

These estimators are unbiased estimators of the directions D_y, D_v, and D_x. The SABA algorithm corresponds to Algorithm 5 where we use Equations (4.6)a (4.6)b and (4.6)c as update directions. When taking a step size $\gamma^t = 0$ in the outer problem, hereby stopping progress in x, we recover the iterations of the SAGA algorithm on the inner problem. In practice, the sum in S is computed by doing a rolling average of the past gradients computed. More precisely, let us denote

$$A_t = \frac{1}{n}\sum_{i'=1}^{n} \phi_{i'}(w_{i'}^t) \ ,$$

the average of the memory at time t. To get A_{t+1}, instead of computing the summing all the gradients stored, which has $\mathcal{O}(n)$ computational complexity, we do

$$A_{t+1} = A_t + \frac{1}{n}(\phi_i(w_i^{t+1}) - \phi_i(w_i^t)) \ ,$$

which is equivalent mathematically but has $\mathcal{O}(1)$ computational complexity.

Moreover, the quantities $\phi_i(w_i^t)$ are stored rather than recomputed: the cost of computing the SABA directions is the same as that of SOBA. It requires an additional memory for the five quantities, of total

size $n \times p + (n+m) \times (p+d)$ floats. Note that we can reduce this load by using larger batch sizes since in this case, we store average over batches rather than individual gradients. Thus, if b_{in} and b_{out} are respectively the inner and the outer batch sizes, the memory load is reduced to $n_b \times p + (n_b + m_b) \times (p \times d)$ with $n_b = \lceil \frac{n}{b_{\text{inn}}} \rceil$ and $m_b = \lceil \frac{m}{b_{\text{out}}} \rceil$ which are smaller than the number of samples. This memory load can also be reduced in specific cases, for instance when g and f correspond to linear models, where the individual gradients and Hessian-vector products are proportional to the samples. In this case, we only store the proportionality ratio, reducing the memory load to $3n + 2m$ floats. Like for SOBA, the computations of the new quantities $\phi_i(w_i^{t+1})$ are done in parallel, thus benefiting from hardware acceleration and shared computations. Despite this memory load, using SAGA-like variance reduction instead of STORM as done by Li et al. (2022), Yang et al. (2021), and Khanduri et al. (2021) has the advantage of bringing the variance of the estimate directions to zero, enabling faster $\mathcal{O}(\frac{1}{T})$ convergence.

In the next section, we show that SABA is fast. It essentially has the same properties as SAGA: despite being stochastic, it converges with fixed step size and reaches the same rate of convergence as gradient descent on Φ.

4.3 Theoretical analysis

In this section, we provide convergence rates of SOBA and SABA under some classical assumptions. Note that, unlike most of the stochastic bilevel optimization papers, we work in a finite sample setting rather than the more general expectation setting. Actually, SABA does not make any sense for functions that don't have a finite-sum structure. However, we stress that SOBA could be studied in a more general setting to obtain the same bounds as here. Also, the finite sum setting is still interesting since doing empirical risk minimization is very common in practice in machine learning. The proofs and the constants in big-$\mathcal{O}$ are deferred in Section A.1.

4.3.1 Background and assumptions

We start by stating some regularity assumptions on the functions f and g.

Assumption 4.1. *The function f is twice differentiable. The derivatives ∇f and $\nabla^2 f$ are Lipschitz continuous in (x, y) with respective Lipschitz constants $L_{f,1}$ and $L_{f,2}$.*

Note that the above assumption is typically verified in the machine learning context, *e.g.,* when f is the ordinary least squares (OLS) loss or the logistic loss.

Assumption 4.2. *The function g is three times continuously differentiable on $\mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$. For any $x \in \mathbb{R}^{d_x}$, $g(x, \cdot)$ is μ_g-strongly convex. The derivatives ∇g, $\nabla^2 g$ and $\nabla^3 g$ are Lipschitz continuous in with respective Lipschitz constants $L_{g,1}$, $L_{g,2}$ and $L_{g,3}$.*

Strong convexity and smoothness with respect to y of g are verified when g is a regularized least-squares/logistic regression with a full rank design matrix when the data is not separable for the logistic regression. Moreover, the strong convexity ensures the existence and uniqueness of the inner optimization problem for any $x \in \mathbb{R}^{d_x}$.

Assumption 4.3. *There exists $C_f > 0$ such that for any x we have $\|\nabla_y f(x, y^*(x))\| \leq C_f$.*

This assumption, combined with the strong convexity of $g(x, \cdot)$, shows boundedness of v^*. This assumption holds, for instance, in the case of hyperparameters selection for a Ridge regression problem. Note that in Assumptions 4.1 and 4.2, we assume more regularity of f and g than in stochastic bilevel optimization literature (see for instance Ghadimi and Wang (2018), Hong et al. (2023), Ji et al. (2021) or Arbel and Mairal (2022a)). It is necessary to get the smoothness of v^*, which will allow to adapt the proof of Chen et al. (2021) and get tight convergence rates. The following lemma gives us some

smoothness properties of the considered directions that will be useful to derive convergence rates of our methods.

Lemma 4.1. *Under the Assumptions 4.1 to 4.3, there exist constants L_y, L_v and L_x such that*

$$\begin{aligned}\|D_y(y,v,x)\|^2 &\leq L_y^2\|y-y^*(x)\|^2\\ \|D_v(y,v,x)\|^2 &\leq L_v^2(\|y-y^*(x)\|^2+\|v-v^*(x)\|^2)\\ \|D_x(y,v,x)-\nabla\Phi(x)\|^2 &\leq L_x^2(\|y-y^*(x)\|^2+\|v-v^*(x)\|^2)\ .\end{aligned}$$

Proof. Let $(y,v,x)\in\mathbb{R}^{d_y}\times\mathbb{R}^{d_y}\times\mathbb{R}^{d_x}$. Using the fact that $\nabla_y g(x,y^*(x))=0$ and the $L_{g,1}$-smoothness of $g(x,\cdot)$, we have

$$\|D_y(y,v,x)\|^2=\|\nabla_y g(x,y)-\nabla_y g(x,y^*(x))\|^2\leq L_{g,1}^2\|y-y^*(x)\|^2\ .$$

For D_v, since $\nabla^2_{yy}g(x,y^*(x))v^*(x)=-\nabla_y f(x,y^*(x))$, we write

$$\begin{aligned}\|D_v\| &= \|(\nabla^2_{yy}g(x,y)v+\nabla_y f(x,y))-(\nabla^2_{yy}g(x,y^*(x))v^*(x)+\nabla_y f(x,y^*(x)))\|\\ &\leq \|[\nabla^2_{yy}g(x,y)-\nabla^2_{yy}g(x,y^*(x))]v^*(x)\|+\|\nabla^2_{yy}g(x,y)[v-v^*(x)]\|\\ &\quad+\|\nabla_y f(x,y)-\nabla_y f(x,y^*(x))\|\ .\end{aligned}$$

For the first term, we use the Lipschitz continuity of $\nabla^2_{yy}g$:

$$\|[\nabla^2_{yy}g(x,y)-\nabla^2_{yy}g(x,y^*(x))]v^*(x)\|\leq L_{g,2}\|y-y^*(x)\|\|v^*(x)\|\ .$$

Since g in μ_g-strongly convex, $\nabla_y f(\,\cdot\,,y^*(\cdot))$ is bounded and $v^*(x)=-[\nabla^2_{yy}g(x,y^*(x))]^{-1}\nabla_y f(x,y^*(x))$, we have

$$\|[\nabla^2_{yy}g(x,y)-\nabla^2_{yy}g(x,y^*(x))]v^*(x)\|\leq\frac{L_{g,2}C_f}{\mu_g}\|y-y^*(x)\|\ . \tag{4.7}$$

For the second term, we use the $L_{g,1}$-smoothness of $g(x,\cdot)$ and for the third term, we use the $L_{f,1}$-smoothness of f and we finally get

$$\|D_v\|\leq\left(\frac{L_{g,2}C_f}{\mu_g}+L_{f,1}\right)\|y-y^*(x)\|+L_{g,1}\|v-v^*(x)\|\ .$$

Then, taking $L_v=\sqrt{2}\max\left(\frac{L_{g,2}C_f}{\mu_g}+L^F,L_{g,1}\right)$, we get

$$\|D_v(y,v,x)\|^2\leq L_v^2(\|y-y^*(x)\|^2+\|v-v^*(x)\|^2)\ .$$

For $D_x(y,v,x)-\nabla\Phi(x)$ we start by writing

$$\begin{aligned}\|D_x(y,v,x)-\nabla\Phi(x)\| &\leq \|\nabla_x f(x,y)-\nabla_x f(x,y^*(x))\|+\|\nabla^2_{xy}g(x,y)v-\nabla^2_{xy}g(x,y^*(x))v^*(x)\|\\ &\leq \|\nabla_x f(x,y)-\nabla_x f(x,y^*(x))\|+\|\nabla^2_{xy}g(x,y)\|\|v-v^*(x)\|\\ &\quad+\|v^*(x)\|\|\nabla^2_{xy}g(x,y)-\nabla^2_{xy}g(x,y^*(x))\|\ .\end{aligned}$$

We bound the first term using the fact that $\nabla_x f$ is $L_{f,1}$-Lipschitz continuous. For the second term, the fact that $\nabla^2_{xy}G$ is bounded thanks to the Lipschitz continuity of $\nabla_y g(\,\cdot\,,y)$. For the third term, we use that $\nabla^2_{xy}g(x,\cdot)$ is $L_{g,2}$-Lipschitz continuous and the same derivation as Equation (4.7). We finally get

$$\|D_x-\nabla\Phi(x)\|\leq\left(L_{f,1}+\frac{C_f L_{g,2}}{\mu_g}\right)\|y-y^*(x)\|+L_{g,1}\|v-v^*(x)\|\ .$$

Taking $L_x = \sqrt{2}\max\left(L_{f,1} + \frac{C_f L_{g,2}}{\mu_g}, L_{g,1}\right)$ yields

$$\|D_x(y,v,x) - \nabla\Phi(x)\|^2 \leq L_x^2(\|y - y^*(x)\|^2 + \|v - v^*(x)\|^2) \ . \tag{4.8}$$

□

In first-order optimization, a fundamental assumption on the objective function is the smoothness assumption. In the case of vanilla gradient descent applied to a function f, it allows to get a convergence rate of $\|\nabla f(x^t)\|^2$ in $\mathcal{O}(1/T)$, *i.e.* convergence to a stationary point (Nesterov, 2004). We recall that the smoothness of Φ is provided in Proposition 3.2

As usual with the analysis of stochastic methods, we define the expected norms of the directions $V_y^t = \mathbb{E}[\|D_y^t\|^2]$, $V_v^t = \mathbb{E}[\|D_v^t\|^2]$ and $V_x^t = \mathbb{E}[\|D_x^t\|^2]$, where the expectation is taken over the past. Thanks to variance-bias decomposition, they are the sum of the variance of the stochastic direction and the squared-norm of the unbiased direction. For SOBA, we use classical bounds on variances like those found for instance by Hong et al. (2023):

Assumption 4.4. *There exist B_y and B_v such that for all t, $\mathbb{E}_t[\|D_y^t\|^2] \leq B_y^2(1 + \|D_y(y^t,v^t,x^t)\|^2)$ and $\mathbb{E}_t[\|D_v^t\|^2] \leq B_v^2(1 + \|D_v(y^t,v^t,x^t)\|^2)$ where $\mathbb{E}_t$ denotes the expectation conditionally to (y^t,v^t,x^t).*

For SOBA and SABA, we need to bound the expected norm of D_x^t. For SABA, this assumption allows to get the same sample complexity as SAGA for single-level problems.

Assumption 4.5. *There exists B_x such that for all t, $\mathbb{E}_t[\|D_x^t\|^2] \leq B_x^2$.*

Assumptions 4.4 and 4.5 are verified for instance, if all the g_i and $\nabla_y g_i$ have at most quadratic growth, and if f has bounded gradients. They are also verified if the iterates remain in a compact set. Note that we do not assume that g has bounded gradients, as this would contradict its strong convexity. Finally, for the analysis of SABA, we need regularity on each g_i and f_j:

Assumption 4.6. *For all $i \in [n]$ and $j \in [m]$, the functions ∇g_i, ∇f_j, $\nabla^2_{yy} g_i$ and $\nabla^2_{xy} g_i$ are Lipschitz continuous in (x,y).*

4.3.2 Fundamental descent lemmas

Our analysis for SOBA and SABA is based on the control of both

$$\delta_y^t = \mathbb{E}[\|y^t - y^*(x^t)\|^2] \quad \text{and} \quad \delta_v^t = \mathbb{E}[\|v^t - v^*(x^t)\|^2] \ .$$

Strong convexity of g and smoothness of $y^*(x)$ and $v^*(x)$ allow to obtain the following lemma by adapting the proof of Chen et al. (2021). In what follows, we drop the dependency of the step sizes ρ and γ in t for clarity.

Lemma 4.2. *Assume that $\gamma^2 \leq \min\left(\frac{\mu_g L_*^2}{4B_x^2 L_{yx}^2}, \frac{\mu_g L_*^2}{8B_x^2 L_{vx}^2}\right)\rho$. We have:*

$$\delta_y^{t+1} \leq \left(1 - \frac{\rho\mu_g}{4}\right)\delta_y^t + 2\rho^2 V_y^t + \beta_{yx}\gamma^2 V_x^t + \overline{\beta}_{yx}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t,v^t,x^t)\|^2]$$

$$\delta_v^{t+1} \leq \left(1 - \frac{\rho\mu_g}{8}\right)\delta_v^t + \beta_{vy}\rho\delta_y^t + 2\rho^2 V_v^t + \beta_{vx}\gamma^2 V_x^t + \overline{\beta}_{yx}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t,v^t,x^t)\|^2]$$

where $\beta_{yx} = \beta_{vx} = 3L_*^2$, $\overline{\beta}_{yx} = \frac{8L_*^2}{\mu_g}$, $\overline{\beta}_{vx} = \frac{16L_*^2}{\mu_g}$, L_* is the maximum between the Lipschitz constants of y^* and v^* (see Lemma A.1), $\beta_{vy} = \frac{1}{\mu_g^3}(L_{f,1}\mu_g + L_{g,2})^2$, L_{yx} and L_{vx} are respectively the smoothness constants of y^* and v^*.

Proof sketch. The detailed proof is provided in subsection A.1.2.

Inequality for δ_y. We start by expanding the square:

$$\|y^{t+1} - y^*(x^{t+1})\|^2 = \|y^{t+1} - y^*(x^t)\|^2 + \|y^*(x^{t+1}) - y^*(x^t)\|^2 - 2\langle y^{t+1} - y^*(x^t), y^*(x^{t+1}) - y^*(x^t)\rangle$$

Using the strong convexity of $g(x, \cdot)$ and the unbiasedness of D_y^t, we bound the first term by

$$\mathbb{E}[\|y^{t+1} - y^*(x^t)\|^2] \leq (1 - \rho\mu_G)\delta_y^t + \rho^2 \mathbb{E}_t V_y^t \ .$$

The second member is bounded using Lipschitz continuity of y^*:

$$\mathbb{E}[\|y^*(x^{t+1}) - y^*(x^t)\|^2] \leq L_*^2 \gamma^2 V_x^t \ .$$

For the remaining scalar product, we have

$$-2\langle y^{t+1} - y^*(x^t), y^*(x^{t+1}) - y^*(x^t)\rangle = -2[\langle y^t - y^*(x^t), y^*(x^{t+1}) - y^*(x^t)\rangle - \rho\langle D_y^t, y^*(x^{t+1}) - y^*(x^t)\rangle] \ .$$

The second term can be bounded using the Cauchy-Schwarz inequality, the Lipschitz-continuity of y^*, and Young inequality:

$$\mathbb{E}[\rho\langle D_y^t, y^*(x^{t+1}) - y^*(x^t)\rangle] \leq \frac{\rho^2}{2} V_y^t + L_*^2 \frac{\gamma^2}{2} V_x^t \ .$$

For $-2\langle y^t - y^*(x^t), y^*(x^{t+1}) - y^*(x^t)\rangle$, we follow the proof of Chen et al. (2021) which consists in making appear the "unbiased part of $y^*(x^{t+1}) - y^*(x^t)$ by a linear approximation. More precisely, we have

$$\langle y^t - y^*(x^t), y^*(x^{t+1}) - y^*(x^t)\rangle = \underbrace{\langle y^t - y^*(x^t), \mathrm{d}y^*(x^t)(x^{t+1} - x^t)\rangle}_{A} \underbrace{\langle y^t - y^*(x^t), y^*(x^{t+1}) - y^*(x^t) - \mathrm{d}y^*(x^t)(x^{t+1} - x^t)\rangle}_{B} \ .$$

For A, we use the unbiasedness of D_x^t, Cauchy-Schwarz inequality, the Lipschitz continuity of y^* (Lemma A.1) and the identity $ab \leq \eta a^2 + \frac{b^2}{\eta}$ for a suitable choice of $\eta > 0$:

$$\begin{aligned} -2\mathbb{E}[A] &= -2\gamma\mathbb{E}[\langle y^t - y^*(x^t), \mathrm{d}y^*(x^t) D_x(y^t, v^t, x^t)\rangle] \\ &\leq 2L_*\gamma\mathbb{E}[\|y^t - y^*(x^t)\|\|D_x(y^t, v^t, x^t)\|] \\ &\leq \frac{\rho\mu_G}{2}\delta_y^t + \frac{8L_*^2}{\mu_G}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \ . \end{aligned}$$

For B, we use Cauchy-Schwarz inequality, the smoothness of y^* (Lemma A.2), Young inequality and the boundedness of $\mathbb{E}_t[\|D_x^t\|^2]$ to get

$$\begin{aligned} -2\mathbb{E}[B] &\leq 2\mathbb{E}[\|y^t - y^*(x^t)\|\|y^*(x^{t+1}) - y^*(x^t) - \mathrm{d}y^*(x^t)(x^{t+1} - x^t)\|] \\ &\leq L_{yx}\mathbb{E}[\|y^t - y^*(x^t)\|\|x^{t+1} - x^t\|^2] \\ &\leq L_{yx}\nu\mathbb{E}[\|y^t - y^*(x^t)\|^2\|x^{t+1} - x^t\|^2] + \frac{L_{yx}}{\nu}\mathbb{E}[\|x^{t+1} - x^t\|^2] \\ &\leq \frac{L_{yx}^2 B_x^2 \gamma^2}{L_*^2}\delta_y^t + L_*^2\gamma^2 V_x^t \ . \end{aligned}$$

Putting all together yields the final result

Inequality for δ_v. The proof for δ_v is similar to the one for δ_y. The difference leans on the fact that we have to take care of the difference between $v^*(x^t; y^t)$ and $v^*(x^t)$. This difference is controlled by δ_y^t, which explains the appearance of this term in the final inequality. □

We insist that this result is obtained in general for Algorithm 5 with arbitrary unbiased directions. We can, therefore, invoke this lemma for the analysis of both SOBA and SABA.

We use the smoothness of Φ to get the following lemma, which is similar to (Chen et al., 2021, Lemma 1).

Lemma 4.3. *Let $\Phi^t = \mathbb{E}[\Phi(x^t)]$ and $g^t = \mathbb{E}[\|\nabla\Phi(x^t)\|^2]$. We have*

$$\Phi^{t+1} \leq \Phi^t - \frac{\gamma}{2} g^t - \frac{\gamma}{2}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] + \frac{\gamma}{2} L_x^2(\delta_y^t + \delta_v^t) + \frac{L_\Phi}{2}\gamma^2 V_x^t \ . \tag{4.9}$$

Proof. We use smoothness of Φ to get

$$\Phi(x^{t+1}) \leq \Phi(x^t) - \gamma\langle D_x^t, \nabla\Phi(x^t)\rangle + \frac{L_\Phi}{2}\gamma^2\|D_x^t\|^2$$

Then, we take the expectation conditionally to the past iterate and use the unbiasedness of D_x^t to obtain

$$\begin{aligned}
\mathbb{E}_t[\Phi(x^{t+1})] &\leq \Phi(x^t) - \gamma\langle D_x(y^t, v^t, x^t), \nabla\Phi(x^t)\rangle + \frac{L_\Phi}{2}\gamma^2\mathbb{E}_t[\|D_x^t\|^2] \\
&\leq \Phi(x^t) - \frac{\gamma}{2}(\|\nabla\Phi(x^t)\|^2 + \|D_x(y^t, v^t, x^t)\|^2 - \|\nabla\Phi(x^t) - D_x(y^t, v^t, x^t)\|^2) \\
&\quad + \frac{L_\Phi}{2}\gamma^2\mathbb{E}_t[\|D_x^t\|^2]
\end{aligned}$$

where the last inequality comes from the identity $\langle a, b\rangle = \frac{1}{2}(\|a\|^2 + \|b\|^2 - \|a - b\|)^2$. We take the total expectation and use the previous Lemma 4.1 to get

$$\Phi^{t+1} \leq \Phi^t - \frac{\gamma}{2} g^t - \frac{\gamma}{2}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] + \frac{\gamma L_x^2}{2}(\delta_y^t + \delta_v^t) + \frac{L_\Phi}{2}\gamma^2 V_x^t \tag{4.10}$$

□

If $y^t = y^*(x^t)$, $v^t = v^*(x^t)$, that is δ_y, δ_v both cancel and $D_x(y^t, v^t, x^t) = \nabla\Phi(x^t)$, we get an inequality reminiscent of the smoothness inequality for SGD on Φ.

4.3.3 Analysis of SOBA

The analysis of SOBA is based on Lemmas 4.2 and 4.3. We have the following theorem, with fixed step sizes depending on the number of iterations:

Theorem 4.1 (Convergence of SOBA, fixed step size)**.** *Fix an iteration $T > 1$ and assume that Assumptions 4.1 to 4.5 hold. We consider fixed steps $\rho^t = \frac{\bar{\rho}}{\sqrt{T}}$ and $\gamma^t = \xi\rho^t$ with $\bar{\rho}$ and ξ precised in the appendix. Let $(x^t)_{t\geq 1}$ the sequence of outer iterates for SOBA. Then,*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\Phi(x^t)\|^2] = \mathcal{O}(T^{-\frac{1}{2}}) \ .$$

As opposed to Hong et al. (2023), we do not need that the ratio $\frac{\gamma}{\rho}$ goes to 0, which allows getting a complexity (that is, the number of calls to oracles to have an ϵ-stationary solution) in $\mathcal{O}(\epsilon^{-2})$ better than the $\tilde{\mathcal{O}}(\epsilon^{-\frac{5}{2}})$ they have. Also, note that this rate is the same as the one of SGD for non-convex single-level problems.

Proof sketch. The detailed proof is provided in subsection A.1.3.

The proof is a Lyapunov analysis. We consider the following Lyapunov function

$$\mathcal{L}^t = \Phi^t + \phi_y \delta_y^t + \phi_v \delta_v^t \tag{4.11}$$

where ϕ_y and ϕ_v are positive constants to be chosen. For SOBA, the inequalities in Lemma 4.2 take the following form for small enough step sizes

$$\delta_y^{t+1} \leq \left(1 - \frac{\rho\mu_g}{8}\right)\delta_y^t + 2\rho^2 B_y^2 + \beta_{yx}\gamma^2 B_x^2 + \overline{\beta}_{yx}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \tag{4.12}$$

$$\delta_v^{t+1} \leq \left(1 - \frac{\rho\mu_g}{16}\right)\delta_v^t + 2\beta_{vy}\rho\delta_y^t + 2\rho^2 B_v^2 + \beta_{vx}\gamma^2 B_x^2 + \overline{\beta}_{vx}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \ . \tag{4.13}$$

Let $\phi_y' = \phi_y\frac{\gamma}{\rho}$ and $\phi_v' = \phi_v\frac{\gamma}{\rho}$. From Lemma 4.3 and using Equations (4.12) and (4.13) with small enough constants ϕ_y', ϕ_v', γ and ρ yields

$$\mathcal{L}^{t+1} - \mathcal{L}^t \leq -\frac{\gamma}{2}g^t + \frac{L_\Phi}{2}B_x^2\gamma^2 + \left(\phi_y'\beta_{yx} + \phi_v'\beta_{vx}\right)B_x^2\rho\gamma + 2\left(\phi_y' B_y^2 + \phi_v' B_v^2\right)\frac{\rho^3}{\gamma} \ . \tag{4.14}$$

Then, summing and telescoping yields the result. □

We obtain a similar rate using decreasing step sizes:

Theorem 4.2 (Convergence of SOBA, decreasing step size)**.** *Assume that Assumptions 4.1 to 4.5 hold. We consider steps $\rho^t = \overline{\rho}t^{-\frac{1}{2}}$ and $\gamma^t = \xi\rho$. Let x^t the sequence of outer iterates for SOBA. Then,*

$$\inf_{t\leq T}\mathbb{E}[\|\nabla\Phi(x^t)\|^2] = \mathcal{O}(\log(T)T^{-\frac{1}{2}}) \ .$$

Proof sketch. The detailed proof is provided in subsection A.1.4.

With the Lyapunov function given by Equation (4.11), Equation (4.14) is still valid for suitable choices of the different constants.

We lower bound $\sum_{t=1}^T \gamma^t g^t$ by $\inf_{t\in[T]} g^t\gamma^T$ and we upper bound $\sum_{t=1}^T(\gamma^t)^2$ by $C(1+\log(T))$ for some constant $C > 0$.

Then, summing and rearranging Equation (4.14) and using the previous bounds give the result. □

As for SGD, SOBA suffers from the need of decreasing step sizes to get actual convergence because of the variance of the estimation on each directions.

On the other hand, the analysis of SABA leverages the dynamic of all three variables, resulting in fast convergence with fixed step sizes.

4.3.4 SABA: a stochastic method with optimal rates

In what follows, we denote $N = n + m$ the total number of samples. The following theorem shows the $\mathcal{O}(N^{\frac{2}{3}}T^{-1})$ convergence for the SABA algorithm in the general case where we only assume smoothness of Φ. Our analysis of SABA is inspired by the analysis of single-level SAGA by Reddi et al. (2016).

Theorem 4.3 (Convergence of SABA, smooth case)**.** *Assume that Assumptions 4.1 to 4.3 and 4.5 to 4.6 hold. We suppose $\rho = \rho' N^{-\frac{2}{3}}$ and $\gamma = \xi\rho$, where ρ' and ξ depend only on f and g and are specified in appendix. Let x^t the iterates of SABA. Then,*

$$\frac{1}{T}\sum_{t=1}^T \mathbb{E}[\|\nabla\Phi(x^t)\|^2] = O\left(N^{\frac{2}{3}}T^{-1}\right) \ .$$

Proof sketch. The detailed proof is provided in subsection A.1.5.

In short, the main difference with the SOBA proof is that we have to control the variance of the estimate of the directions D_y^t, D_v^t and D_x^t. This is done by controlling the distance from the memory to the current variables. We denote for indices $i \in [n]$ and $j \in [m]$ we denote (y_i^t, v_i^t, x_i^t) and $(y_j'^t, v_j'^t, x_j'^t)$ the memories for each variable corresponding respectively to calls to g and f. We define

$$E_y^t = \frac{1}{n}\sum_{i=1}^n \mathbb{E}[\|y^t - y_i^t\|^2], \quad E_v^t = \frac{1}{n}\sum_{i=1}^n \mathbb{E}[\|v^t - v_i^t\|^2], \quad E_x^t = \frac{1}{n}\sum_{i=1}^n \mathbb{E}[\|x^t - x_i^t\|^2]$$
$$E_y'^t = \frac{1}{m}\sum_{j=1}^m \mathbb{E}[\|y^t - y_j'^t\|^2], \quad E_v'^t = \frac{1}{m}\sum_{j=1}^m \mathbb{E}[\|v^t - v_j'^t\|^2], \quad E_x'^t = \frac{1}{m}\sum_{j=1}^m \mathbb{E}[\|x^t - x_j'^t\|^2] \ .$$

These quantify the error between the iterates and their respective memories. We show in Lemma A.5 that we have the inequality

$$E_y^{t+1} \leq \left(1 - \frac{1}{2n}\right) E_y^t + \rho^2 \mathbb{E}[\|D_y^t\|^2] + 2n\rho^2 \mathbb{E}[\|D_y(y^t, v^t, x^t)\|^2]$$

and similar inequalities for E_v^t, E_x^t, $E_y'^t$, $E_v'^t$ and $E_x'^t$.

In a second step, we prove in Lemma A.6 the control of the variance by the memory error and the quantities δ_y^t and δ_v^t:

$$\begin{aligned}
\mathbb{E}[\|D_y^t\|^2] &\leq 2L_y^2 \delta_y^t + 2L_y' S^t \ , \\
\mathbb{E}[\|D_v^t\|^2] &\leq 2(L_v^2 + L_y'')(\delta_y^t + \delta_v^t) + 2L_v' S^t \ , \\
\mathbb{E}[\|D_x^t\|^2] &\leq 2\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] + 2L_x''(\delta_y^t + \delta_v^t) + 2L_x' S^t
\end{aligned}$$

where $S^t = E_y^t + E_v^t + E_x^t + E_y'^t + E_v'^t + E_x'^t$ and $L_y, L_y', L_y'', L_v, L_v', L_v'', L_x, L_x'$ are constants.

From the previous inequalities, we derive a control on the sum of the memory errors S^t

$$S^{t+1} \leq \left(1 - \frac{\Gamma}{2}\right) S^t + \beta_{sy}\rho^2 \delta_y^t + \beta_{sv}\rho^2 + P\gamma^2 \mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \ . \tag{4.15}$$

Then, the control of the variances enables us to get a modified version of the descent lemmas 4.2 and 4.3

$$\delta_y^{t+1} \leq \left(1 - \frac{\rho\mu_g}{8}\right) \delta_y^t + 2L_x''\beta_{yx}\gamma^2\delta_v^t + 5L_y'\rho^2 S^t + 2\overline{\beta}_{yx}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \ , \tag{4.16}$$

$$\delta_v^{t+1} \leq \left(1 - \frac{\rho\mu_g}{16}\right) \delta_v^t + 3\beta_{vy}\rho\delta_y^t + 5L_v'\rho^2 S^t + 2\overline{\beta}_{vx}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \ , \tag{4.17}$$

$$\Phi^{t+1} \leq \Phi^t - \frac{\gamma}{2}g^t - \frac{\gamma}{4}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] + L_x^2\gamma(\delta_y^t + \delta_v^t) + L_\Phi L_x'\gamma^2 S^t \ . \tag{4.18}$$

Finally, we define the Lyapunov function

$$\mathcal{L}^t = \Phi^t + \phi_s S^t + \phi_y \delta_y^t + \phi_v \delta_v^t \ .$$

We use Equations (4.15) to (4.18) to upper bound $\mathcal{L}^{t+1}$ in function of the quantities S^t, δ_y^t, δ_v^t, Φ^t, g^t and $\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2]$. Then, we show that under appropriate choices of step sizes and constants ϕ_s, ϕ_y, ϕ_v, we have

$$\mathcal{L}^{t+1} - \mathcal{L}^t \leq -\frac{\gamma}{2}g^t \ .$$

Finally, summing, telescoping and rearranging yield the result. □

Note that the step sizes are constant with respect to the time, but they scale with $N^{-\frac{2}{3}}$. As a consequence, the sample complexity is $O(N^{\frac{2}{3}}\epsilon^{-1})$ which is analogous to the one of SAGA for non-convex single-level problems (Reddi et al., 2016). This is better than the sample complexity of Algorithm 5 with full batch directions, which is $\mathcal{O}(N\epsilon^{-1})$. Hence, with SABA, we get the best of both worlds: the stochasticity makes the scaling in N of the sample complexity going from N in full batch mode to $N^{\frac{2}{3}}$ for SABA, and the variance reduction makes the scaling in ϵ goes from ϵ^{-2} for SOBA to ϵ^{-1} for SABA. Our experiments in Section 4.4 confirm this gain.

Furthermore, if we assume that Φ satisfies a Polyak-Łojasiewicz (PL) inequality, we recover linear convergence. Recall that Φ has the PL property if there exists $\mu_\Phi > 0$ such that for all $x \in \mathbb{R}^{d_x}$,

$$\frac{1}{2}\|\nabla\Phi(x)\|^2 \geq \mu_\Phi(\Phi(x) - \Phi^*)$$

with Φ^* the minimum of Φ.

Theorem 4.4 (Convergence of SABA, PL case)**.** *Assume that Φ satisfies the PL inequality and that Assumptions 4.1 to 4.3 and 4.5 to 4.6 hold. We suppose $\rho = \rho' N^{-\frac{2}{3}}$ and $\gamma = \xi\rho' N^{-1}$, where ρ' and ξ depend only on f and g and are specified in appendix. Let x^t the iterates of SABA and $c' \triangleq \min\left(\mu_\Phi, \frac{1}{16P'}\right)$ with P' specified in the appendix. Then,*

$$\mathbb{E}[\Phi^T] - \Phi^* = (1 - c'\gamma)^T(\Phi^0 - \Phi^* + C^0)$$

where C^0 is a constant specified in appendix that depends on the initialization of y, v, x and memory.

Proof sketch. The detailed proof is provided in subsection A.1.6.

The proof is similar to that of the previous theorem: we find coefficients ϕ_s, ϕ_y, ϕ_v such that

$$\mathcal{L}^t = \Phi^t + \phi_s S^t + \phi_y \delta_y^t + \phi_v \delta_v^t$$

satisfies the inequality

$$\mathcal{L}^{t+1} \leq (1 - c'\gamma)\mathcal{L}^t \ ,$$

which is then unrolled. □

Note that in the case where we initialize y and v with $y^0 = y^*(x^0)$, $v^0 = v^*(x^0)$, and the memories $w_i^0 = w^0$, $\tilde{w}_j^0 = w^0$ for all i, j, the constant C^0 cancels and the bound simplifies to

$$\mathbb{E}[\Phi(x^T)] - \Phi^* \leq (1 - c'\gamma)^T(\Phi(x^0) - \Phi^*) \ .$$

Just like classical variance reduction methods in single-level optimization, this theorem shows that our method achieves linear convergence under PL assumption on the value function. To the best of our knowledge, our method is the first stochastic bilevel optimization method that enjoys such property. We note that the PL hypothesis is more general than μ_Φ-strong convexity of Φ – it is a necessary condition for strong convexity.

We see here the importance of *global* variance reduction. Indeed, using variance reduction only on y and SGD on x would lead to sub-linear convergence in x. This would be the case even with a perfect estimation of $y^*(x)$. Similarly, using variance reduction only on x and SGD on y would lead to sub-linear convergence in y, and hence in x. Using global variance reduction with respect to each variable, as we propose here, is the only way to achieve linear convergence. We now turn to experiments, where we find that our method is also promising from a practical point of view.

4.4 Experiments

Here we compare the performances of SOBA and SABA with competitor methods on different tasks.

The different methods being compared are stocBiO (Ji et al., 2021), AmiGO (Arbel and Mairal, 2022a), FSLA (Li et al., 2022), MRBO (Yang et al., 2021), TTSA (Hong et al., 2023), BSA (Ghadimi and Wang, 2018) and SUSTAIN (Khanduri et al., 2021). [1]

4.4.1 Hyperparameters selection

The first task we perform is hyperparameters selection to choose regularization parameters on ℓ^2 logistic regression. Let us denote $((d_i^{\text{train}}, y_i^{\text{train}}))_{1\leq i\leq n}$ and $((d_i^{\text{val}}, y_i^{\text{val}}))_{1\leq i\leq m}$ the training and the validation sets. In this case, the inner variable θ corresponds to the parameters of the model, and the outer variable λ to the regularization. The functions f and g of the problem (4.1) are the logistic loss, with ℓ^2 penalty for g, that is to say

$$f(\lambda, \theta) = \frac{1}{m}\sum_{i=1}^{m} \varphi(y_i^{\text{val}} \langle d_i^{\text{val}}, \theta\rangle)$$

and

$$g(\lambda, \theta) = \frac{1}{n}\sum_{i=1}^{n} \varphi(y_i^{\text{train}} \langle d_i^{\text{train}}, \theta\rangle) + \frac{1}{2}\sum_{k=1}^{p} e^{\lambda_k}\theta_k^2$$

where $\varphi(u) = \log(1+e^{-u})$. We fit a binary classification model on the `IJCNN1`[2] dataset. Here, $n = 49\,990$, $m = 91\,701$ and $p = 22$. Note that the parametrization in e^λ of the penalty instead of λ can be surprising at first glance, but it is classical in the bilevel optimization literature (Pedregosa, 2016; Ji et al., 2021; Grazzi et al., 2021) because it avoids positivity constraints on λ.

The suboptimality gap is plotted in figure 4.2 for each method. The lowest values are reached by SABA. Moreover, SABA is the only single-loop method that reaches a suboptimality below 10^{-3}. SOBA reaches a quite high final value but slightly better than TTSA and FSLA. The gap between SOBA and SABA highlights the benefits of variance reduction: it gives us a lower plateau and the fixed step sizes enable faster convergence.

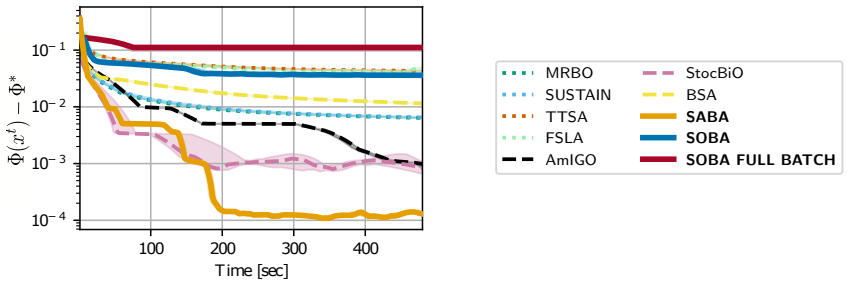


Figure 4.2: Comparison of SOBA and SABA with other stochastic bilevel optimization methods. For each algorithm, we plot the median performance over 10 runs. The shaded area corresponds to the performances between the 20% and the 80% percentiles. We observe that SABA achieves the best performance. The dashed lines are for one-loop competitor methods, the dotted lines are for two-loop methods, and the solid lines are the proposed methods.

4.4.2 Data hypercleaning

The second task we perform is data hypercleaning introduced by Franceschi et al. (2017) on the `MNIST`[3] dataset. The data is patitioned into a training set $(d_i^{\text{train}}, y_i^{\text{train}})$, a validation set $(d_i^{\text{val}}, y_i^{\text{val}})$, and a test

[1] The code of the benchmark is available at `https://github.com/benchopt/benchmark_bilevel` and the results are displayed in `https://benchopt.github.io/results/benchmark_bilevel.html`.

[2] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`

[3] `http://yann.lecun.com/exdb/mnist/`

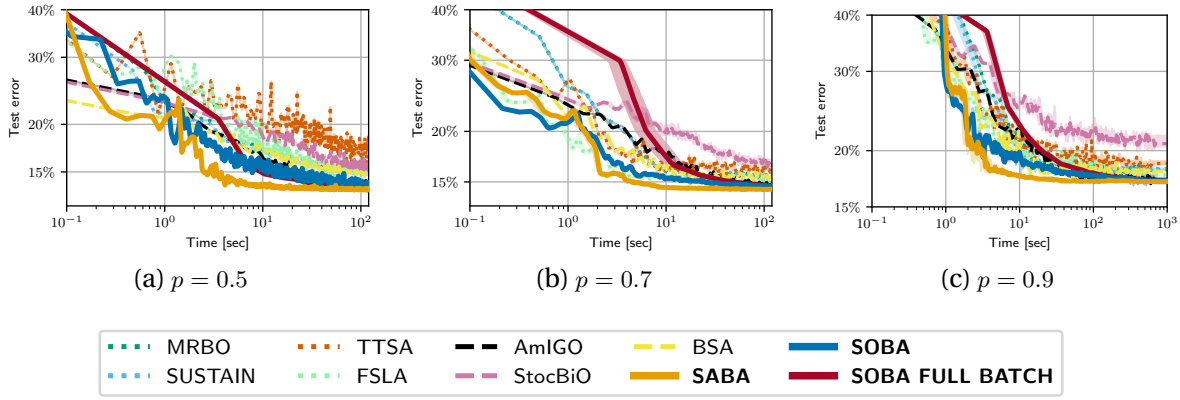
set. The training set contains 20000 samples, the validation set 5000 samples and the test set 10000 samples. The targets y take values in $\{0, \dots, 9\}$ and the samples x are in dimension 784. Each sample in the training set is *corrupted* with probability p: a sample is corrupted when we replace its label y_i by a random label in $\{0, \dots, 9\}$. Samples in the validation and test sets are not corrupted. The goal of datacleaning is to train a multinomial logistic regression on the train set and learn a weight per training sample, that should go to 0 for corrupted samples. This is formalized by the bilevel optimization problem (4.1) with

$$f(\lambda, \theta) = \frac{1}{m} \sum_{i=1}^{m} \ell(\theta d_i^{\text{val}}, y_i^{\text{val}})$$

and

$$f(\lambda, \theta) = \frac{1}{n} \sum_{i=1}^{n} \sigma(\lambda_i) \ell(\theta d_i^{\text{train}}, y_i^{\text{train}}) + C_r \|\theta\|^2$$

where ℓ is the cross entropy loss and σ is the sigmoid function. The inner variable θ is a matrix of size 10×784, and the outer variable λ is a vector in dimension $n_{\text{train}} = 20000$.



(a) $p = 0.5$ (b) $p = 0.7$ (c) $p = 0.9$

Figure 4.3: Datacleaning experiment, with different corruption probability (higher means that more data are contaminated). For each plot, we report the median test error over 10 runs. The shaded area corresponds to the performance between the 20% and the 80% percentiles.

For the estimated parameters θ during optimization, we report in figure 4.3 the test error, *i.e.*, the percent of wrong predictions on the testing data. We use for this experiment a corruption probability $p \in \{0.5, 0.7, 0.9\}$. In general, the error decreases quickly until it reaches a final value. We observe that our method SABA outperforms all the other methods by reaching faster its smallest error, which is smaller than the ones of the other methods. For SOBA, it reaches a lower final error than stocBiO and BSA. Overall, we find that among all methods, even those that implement variance reduction (that is FSLA, MRBO, SUSTAIN, SABA), SABA is the one that demonstrates the best empirical performance.

4.4.3 Implementation details

All the experiments are performed with `Python`, using the package `Benchopt` (Moreau et al., 2022) and Numba (Lam et al., 2015) for fast implementation of stochastic methods. For each problem, we use oracles for a function given function f that $(f(x, y), \nabla_y f(z, x), \nabla^2_{yy} f(x, y)v, \nabla^2_{21} f(x, y)v)$ avoiding duplicate computation of intermediate results for these quantities.

We find that using mini-batches instead of individual samples to compute the stochastic estimates allowed for much faster computations, thanks to hardware acceleration and vectorization of the computations. We use continuous batches to avoid random memory access that slows down the computations. Concretely, if i_b is the index of the current batch and B is the batch size, the indices of the corresponding samples are those in the set $\{i_b \times B, \dots, (i_b + 1) \times B - 1\}$. By doing so, the samples in the same batch are contiguous in memory, which facilitates access. We use a batch size of 64 in all experiments.

For the methods involving an inner loop (stocBiO, BSA, AmIGO), we perform 10 inner steps at each outer iteration as proposed in the papers which introduced these methods. For the approximate Hessian vector product, we perform 10 steps per outer iteration for each methods using HIA (BSA, TTSA, SUSTAIN), SHIA (MRBO, stocBiO) or SGD (AmIGO) for the inversion of the linear system.

For the step sizes, they all have the form $\rho^t = \alpha/t^a$ and $\gamma^t = \beta/t^b$. For the pair of exponents (a, b), we choose the theoretical one from the original papers, that is $(1/2, 1/2)$ for BSA and FSLA, $(1/3, 1/3)$ for MRBO and SUSTAIN, $(0, 0)$ for SABA, AmIGO and stocBiO, $(2/5, 3/5)$ for TTSA and SOBA. For (α, β), we perform a grid search, and we keep for each method, the pair (α, β) that gives the lowest value of Φ (for the hyperparameters) or the lowest test error (for the data cleaning task) in median over 10 runs for each possible pair. When we use HIA or SHIA for the Hessian inversion, we set $\eta = \alpha$ since the Hessian inversion problem has the same conditioning as the inner optimization problem. For the STORM's momentum parameter in MRBO and SUSTAIN, we take $0.5/t^{2/3}$.

For the grid search of the hyperparameter selection experiment, we search α in a set of 9 values between 2^{-5} and 2^3 spaced on a log scale. For β, we choose r in a set of 7 values between 10^{-2} and 10 spaced on a logarithmic scale and we set $\beta = \frac{\alpha}{r}$. For this experiment, we use Just-In-Time (JIT) compilation thanks to the package Numba (Lam et al., 2015), to decrease the python overhead in the iteration loop. To evaluate the value function Φ, we use L-BFGS (Liu and Nocedal, 1989) to solve compute $y^*(x^t)$ and then evaluate the function $\Phi(x^t) = f(x^t, y^*(x^t))$.

For the datacleaning experiment, the parameter α is picked in a set of 11 numbers between 10^{-3} and 100 spaced on a logarithmic scale. For β, we choose r in a set of 11 values between 10^{-5} and 1 spaced on a logarithmic scale and we set $\beta = \frac{\alpha}{r}$. For the regularization parameter C_r, we choose $C_r = 0.2$ after a manual search in order to get the best final test accuracy. Note that in this case, we could not use JIT from Numba since at the moment of the experiment, the softmax function coming from Scipy was not compatible with Numba.

4.5 Conclusion

In this chapter, we have presented a framework for bilevel optimization that enables the straightforward development of stochastic algorithms. The gist of our framework is that the directions in Equations (4.4)a, (4.4)b, and (4.4)c are all written as simple sums of sample derivatives. We leveraged this fact to propose SOBA, an extension of SGD to our framework, and SABA, an extension of SAGA to our framework, which both achieve similar convergence rates as their single-level counterparts. Finally, we think that our framework opens a large panel of potential methods for stochastic bilevel optimization involving techniques of extrapolation, variance reduction, momentum, and so on.

CHAPTER A

APPENDIX TO A FRAMEWORK FOR BILEVEL OPTIMIZATION THAT ENABLES STOCHASTIC AND GLOBAL VARIANCE REDUCTION ALGORITHMS

Section A.1 contains the proofs of the analysis of SOBA and SABA that are not include in the main text. Section A.2 provides convergence rates of SOBA and SABA algorithms if we only assume that the outer function f is differentiable and smooth (instead of twice differentiable with Lipschitz continuous Hessians and gradients) and the inner function g is twice differentiable with Lipschitz continuous derivatives (instead of three times differentiable).

A.1 Proofs

A.1.1 Lemmas on the regularity of y^* and v^*

We start by showing the Lipschitz continuity of y^* and v^*.

Lemma A.1. *There exists a constant $L_* > 0$ such that for any $x_1, x_2 \in \mathbb{R}^{d_x}$ we have*

$$\|y^*(x_1) - y^*(x_2)\| \leq L_* \|x_1 - x_2\|, \quad \|v^*(x_1) - v^*(x_2)\| \leq L_* \|x_1 - x_2\| \ .$$

Proof. Let $x \in \mathbb{R}^{d_x}$. The Jacobian of y^* is given by $\mathrm{d}y^*(x) = -[\nabla^2_{yy} g(x, y^*(x))]^{-1} \nabla^2_{yx} g(x, y^*(x))$. Thanks to the μ_g-strong convexity of g and the fact that $\nabla^2_{21} G$ is bounded, we have $\|\mathrm{d}y^*(x)\| \leq \frac{L_{g,1}}{\mu_g}$. Thus, y^* is Lipschitz continuous.

For $\|v^*(x_1) - v^*(x_2)\|$, we start from the definition v^*:

$$\begin{aligned}
\|v^*(x_1) - v^*(x_2)\| &= \|[\nabla^2_{yy} g(x_1, y^*(x_1))]^{-1} \nabla_y f(x_1, y^*(x_1)) - [\nabla^2_{yy} g(x_2, y^*(x_2))]^{-1} \nabla_y f(x_2, y^*(x_2))\| \\
&\leq \|([\nabla^2_{yy} g(x_1, y^*(x_1))]^{-1} - [\nabla^2_{yy} g(x_2, y^*(x_2))]^{-1} \nabla_y f(x_1, y^*(x_1))\| \\
&\quad + \|[\nabla^2_{yy} g(x_2, y^*(x_2))]^{-1} (\nabla_y f(x_2, y^*(x_2)) - \nabla_y f(x_1, y^*(x_1)))\| \ .
\end{aligned}$$

For the first term, we use that for any invertible matrix A and B we have $A^{-1} - B^{-1} = A^{-1}(B - A)B^{-1}$ to get

$$\begin{aligned}\|[\nabla^2_{yy}g(x_1,y^*(x_1))]^{-1}-\nabla^2_{yy}g(x_2,y^*(x_2))]^{-1}\| &= \|[\nabla^2_{yy}g(x_1,y^*(x_1))]^{-1}(\nabla^2_{yy}g(x_2,y^*(x_2))]- \\ &\quad \nabla^2_{yy}g(x_1,y^*(x_1))])[\nabla^2_{yy}g(x_2,y^*(x_2))]^{-1}\| \\ &\leq \frac{1}{\mu_g^2}\|\nabla^2_{yy}g(x_1,y^*(x_1))-\nabla^2_{yy}g(x_2,y^*(x_2))\| \\ &\leq \frac{L_{g,2}}{\mu_g^2}\|(x_1,y^*(x_1))-(y^*(x_2),x_2)\| \\ &\leq \frac{L_{g,2}}{\mu_g^2}[\|y^*(x_1)-y^*(x_2)\|+\|x_1-x_2\|] \\ &\leq \frac{L_{g,2}}{\mu_g^2}\left[1+\frac{L_{g,1}}{\mu_g}\right]\|x_1-x_2\| \ .\end{aligned}$$

And then, since $\nabla_y f(\,\cdot\,,y^*(\,\cdot\,))$ is bounded:

$$\left\|([\nabla^2_{yy}g(x_1,y^*(x_1))]^{-1}-[\nabla^2_{yy}g(x_2,y^*(x_2))]^{-1}\nabla_y f(x_1,y^*(x_1))\right\| \leq \frac{C_f L_{g,2}}{\mu_g^2}\left[1+\frac{L_{g,1}}{\mu_g}\right]\|x_1-x_2\| \ .$$

For the second term, the strong convexity of $g(\,\cdot\,,x)$ and the fact that $\nabla_y f$ is Lipschitz continuous lead to

$$\begin{aligned}\|[\nabla^2_{yy}g(x_2,y^*(x_2))]^{-1}(\nabla_y f(x_2,y^*(x_2))-\nabla_y f(x_1,y^*(x_1)))\| &\leq \frac{1}{\mu_g}\|\nabla_y f(x_2,y^*(x_2))-\nabla_y f(x_1,y^*(x_1))\| \\ &\leq \frac{L_{f,1}}{\mu_F}\|(x_1,y^*(x_1))-(y^*(x_2),x_2)\| \\ &\leq \frac{L_{f,1}}{\mu_g}[\|y^*(x_1)-y^*(x_2)\|+\|x_1-x_2\|] \\ &\leq \frac{L_{f,1}}{\mu_g}\left[1+\frac{L_{g,1}}{\mu_g}\right]\|x_1-x_2\| \ .\end{aligned}$$

Then we get

$$\|v^*(x_1)-v^*(x_2)\| \leq \left[\frac{C_f L_{g,2}}{\mu_g^2}\left[1+\frac{L_{g,1}}{\mu_g}\right]+\frac{L_{f,1}}{\mu_g}\left[1+\frac{L_{g,1}}{\mu_g}\right]\right]\|x_1-x_2\| \ .$$

We conclude by setting

$$L_* = \max\left(\frac{L_{g,1}}{\mu_g}, \frac{C_f L_{g,2}}{\mu_g^2}\left[1+\frac{L_{g,1}}{\mu_g}\right]+\frac{L_{f,1}}{\mu_g}\left[1+\frac{L_{g,1}}{\mu_g}\right]\right) \ .$$

□

In what follows, we denote by $\mathbb{E}_t[\,\cdot\,]$ the expectation conditionally on y^t, v^t and x^t.

We have the smoothness property of y^* provided in (Chen et al., 2021, Lemma 2).

Lemma A.2. *Under the Assumptions 4.1, 4.2 and 4.3, the function $y^* : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ is L_{yx}-smooth with*

$$L_{yx} = \frac{L_{g,2}(1+L_*)}{\mu_g}+\frac{L_{g,1}L_{g,2}(1+L_*)}{\mu_g^2} \ .$$

We establish the same result for v^*. To this, we need more regularity on g and f.

Lemma A.3. *The function $v^* : \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ is differentiable and its differential is defined for any $x, \epsilon \in \mathbb{R}^{d_x}$ by:*

$$\begin{aligned}\mathrm{d}v^*(x).\epsilon &= [\nabla^2_{yy}g(x,y^*(x))]^{-1}[\nabla^2_{yy}f(x,y^*(x))\mathrm{d}y^*(x).\epsilon+\nabla^2_{yx}f(x,y^*(x)).\epsilon] \\ &\quad -[\nabla^2_{yy}g(x,y^*(x))]^{-1}[(\nabla^3_{yyy}g(x,y^*(x))|\mathrm{d}y^*(x).\epsilon)+(\nabla^3_{yyx}g(x,y^*(x))|\epsilon)] \\ &\quad \times[\nabla^2_{yy}g(x,y^*(x))]^{-1}\nabla_y f(x,y^*(x))\end{aligned} \tag{A.1}$$

where for any $z, \alpha \in \mathbb{R}^{d_y}$ *and* $x \in \mathbb{R}^{d_x}$, $(\nabla^3_{yyy} g(x,y)|\alpha) \in \mathbb{R}^{d_y \times d_y}$ *is defined by*

$$(\nabla^3_{yyy} g(x,y)|\alpha) = \left[\sum_{k=1}^{p} \frac{\partial^3 g}{\partial y_i \partial y_j \partial y_k}(z,x)\alpha_k\right]_{1\leq i,j\leq d_y}$$

and for any $\beta \in \mathbb{R}^{d_x}$, $(\nabla^3_{yyx} g(x,y)|\beta) \in \mathbb{R}^{d_y \times d_y}$ *is defined by*

$$(\nabla^3_{yyx} g(x,y)|\beta) = \left[\sum_{k=1}^{p} \frac{\partial^3 g}{\partial y_i \partial y_j \partial x_k}(z,x)\beta_k\right]_{1\leq i,j\leq d_y} .$$

Moreover, $\mathrm{d}v^*$ *is* L_{vx}*-Lipschitz continuous.*

Proof. Let $x, \epsilon \in \mathbb{R}^{d_x}$. Using the differentiability of $\nabla^2_{yy} g$, $\nabla_y f$ and of the matrix inversion, we have

$$\begin{aligned}
v^*(x+\epsilon) &= [\nabla^2_{yy} g(y^*(x+\epsilon), x+\epsilon)]^{-1} \nabla_y f(y^*(x+\epsilon), \epsilon) \\
&= [\nabla^2_{yy} g(x, y^*(x)) + (\nabla^3_{yyy} g(x, y^*(x))|\mathrm{d}y^*(x).\epsilon) + (\nabla^3_{yyx} g(x, y^*(x))|\epsilon) + o(\|\epsilon\|)]^{-1} \\
&\quad \times (\nabla_y f(x, y^*(x)) + \nabla^2_{yy} f(x, y^*(x)) \mathrm{d}y^*(x).\epsilon + \nabla^2_{yx} f(x, y^*(x))\epsilon + o(\|\epsilon\|)) \\
&= \big\{ [\nabla^2_{yy} g(x, y^*(x))]^{-1} \\
&\quad - [\nabla^2_{yy} g(x, y^*(x))]^{-1} [(\nabla^3_{yyy} g(x, y^*(x))|\mathrm{d}y^*(x).\epsilon) + (\nabla^3_{yyx} g(x, y^*(x))|\epsilon)] \\
&\quad \times [\nabla^2_{yy} g(x, y^*(x))]^{-1} + o(\|\epsilon\|) \big\} \\
&\quad \times (\nabla_y f(x, y^*(x)) + \nabla^2_{yy} f(x, y^*(x)) \mathrm{d}y^*(x).\epsilon + \nabla^2_{yx} f(x, y^*(x))\epsilon + o(\|\epsilon\|)) \\
&= v^*(x) + [\nabla^2_{yy} g(x, y^*(x))]^{-1} [\nabla^2_{yy} f(x, y^*(x)) \mathrm{d}y^*(x).\epsilon + \nabla^2_{yx} f(x, y^*(x)).\epsilon] \\
&\quad - [\nabla^2_{yy} g(x, y^*(x))]^{-1} [(\nabla^3_{yyy} g(x, y^*(x))|\mathrm{d}y^*(x).\epsilon) + (\nabla^3_{yyx} g(x, y^*(x))|\epsilon)] [\nabla^2_{yy} g(x, y^*(x))]^{-1} \\
&\quad \times \nabla_y f(x, y^*(x)) + o(\|\epsilon\|)
\end{aligned}$$

that proves (A.1). Now, let $x, x', \epsilon \in \mathbb{R}^{d_x}$ with $\|\epsilon\| = 1$. Let us denote

$$A(x,\epsilon) = -[\nabla^2_{yy} g(x, y^*(x))]^{-1} [(\nabla^3_{yyy} g(x, y^*(x))|\mathrm{d}y^*(x).\epsilon) + (\nabla^3_{yyx} g(x, y^*(x))|\epsilon)] [\nabla^2_{yy} g(x, y^*(x))]^{-1}$$

and

$$B(x,\epsilon) = \nabla^2_{yy} f(x, y^*(x)) \mathrm{d}y^*(x).\epsilon + \nabla^2_{yx} f(x, y^*(x))$$

so that $\mathrm{d}v^*(x).\epsilon = [\nabla^2_{yy} g(x, y^*(x))]^{-1} B(x,\epsilon) + A(x,\epsilon)\nabla_y f(x, y^*(x))$. We have

$$\begin{aligned}
(\mathrm{d}v^*(x) - \mathrm{d}v^*(x')).\epsilon &= [\nabla^2_{yy} g(x, y^*(x))]^{-1} B(x,\epsilon) + A(x,\epsilon)\nabla_y f(x, y^*(x)) \\
&\quad - [\nabla^2_{yy} g(x', y^*(x'))]^{-1} B(x',\epsilon) - A(x',\epsilon)\nabla_y f(x', y^*(x')) \\
&= [\nabla^2_{yy} g(x, y^*(x))]^{-1} (B(x,\epsilon) - B(x',\epsilon)) \\
&\quad + ([\nabla^2_{yy} g(x, y^*(x))]^{-1} - [\nabla^2_{yy} g(x', y^*(x'))]^{-1}) B(x',\epsilon) \\
&\quad + A(x,\epsilon)(\nabla_y f(x, y^*(x)) - \nabla_y f(x', y^*(x'))) \\
&\quad + (A(x,\epsilon) - A(x',\epsilon))\nabla_y f(x', y^*(x')) \ .
\end{aligned}$$

We can now bound each term using the regularity assumptions on g and f:

$$\begin{aligned}
\|[\nabla^2_{yy} g(x, y^*(x))]^{-1} (B(x,\epsilon) - B(x',\epsilon))\| &\leq \frac{1}{\mu_g} (\|\nabla^2_{yy} f(x, y^*(x)) \mathrm{d}y^*(x) - \nabla^2_{yy} f(x', y^*(x')) \mathrm{d}y^*(x')\| \\
&\quad + \|\nabla^2_{yx} f(x, y^*(x)) - \nabla^2_{yx} f(x', y^*(x'))\|) \\
&\leq \frac{1}{\mu_g} (\|\nabla^2_{yy} f(x, y^*(x)) - \nabla^2_{yy} f(x', y^*(x'))\| \|\mathrm{d}y^*(x)\| \\
&\quad + \|\mathrm{d}y^*(x) - \mathrm{d}y^*(x')\| \|\nabla^2_{yy} f(x', y^*(x'))\| \\
&\quad + L_{f,2}(\|y^*(x) - y^*(x')\| + \|x - x'\|) \\
&\leq \frac{1}{\mu_g} (L_{f,2} L_* (1 + L_*) + L_{yx} L_{f,1} + L_{f,2}(1 + L_*)) \|x - x'\| \ .
\end{aligned}$$

For the second term:

$$\begin{aligned}\|([\nabla^2_{yy}g(x,y^*(x))]^{-1} - [\nabla^2_{yy}g(x',y^*(x'))]^{-1})B(x',\epsilon)\| & \\ \leq \frac{1}{\mu_g^2}\|\nabla^2_{yy}g(x,y^*(x)) - \nabla^2_{yy}g(x',y^*(x'))\|\|B(x',\epsilon)\| & \\ \leq \frac{1}{\mu_g^2}\|\nabla^2_{yy}g(x,y^*(x)) - \nabla^2_{yy}g(x',y^*(x'))\| & \\ \times (\|\nabla^2_{yy}f(x,y^*(x))\|\|\mathrm{d}y^*(x)\| + \|\nabla^2_{yx}f(x,y^*(x))\|) & \\ \leq \frac{(L_{g,2}+L_{f,1})(L_*+1)}{\mu_g^2}\|x-x'\| \ .\end{aligned}$$

For the third term, we have:

$$\begin{aligned}\|A(x,\epsilon)(\nabla_y f(x,y^*(x)) - \nabla_y f(x',y^*(x')))\| \leq \frac{L_{f,1}(1+L^*)}{\mu_g^2}\|(\nabla^3_{yyy}g(x,y^*(x))|\mathrm{d}y^*(x).\epsilon) & \\ + \quad (\nabla^3_{yyx}g(x,y^*(x))|\epsilon)\|\|x-x'\| & \\ \leq \frac{(L_{f,1}+L_{g,2})(1+L^*)}{\mu_g^2}\|x-x'\| \ .\end{aligned}$$

And finally, for the forth term:

$$\begin{aligned}\|(A(x,\epsilon)-A(x',\epsilon))\nabla_y f(x,y^*(x))\| \leq C_f\{\|[\nabla^2_{yy}g(x,y^*(x))]^{-1}\| & \\ \times \|(\nabla^3_{yyy}g(x,y^*(x))|\mathrm{d}y^*(x).\epsilon) + (\nabla^3_{yyx}g(x,y^*(x))|\epsilon)\| & \\ \times \|[\nabla^2_{yy}g(x,y^*(x))]^{-1} - [\nabla^2_{yy}g(x',y^*(x'))]^{-1}\| & \\ + \|[\nabla^2_{yy}g(x,y^*(x))]^{-1} - [\nabla^2_{yy}g(x',y^*(x'))]^{-1}\| & \\ \times \|(\nabla^3_{yyy}g(x,y^*(x))|\mathrm{d}y^*(x).\epsilon) + (\nabla^3_{yyx}g(x,y^*(x))|\epsilon)\| & \\ \times \|[\nabla^2_{yy}g(x',y^*(x'))]^{-1}\| & \\ + \|[\nabla^2_{yy}g(x',y^*(x'))]^{-1}\|^2 & \\ \times (\|(\nabla^3_{yyy}g(x,y^*(x))|\mathrm{d}y^*(x).\epsilon) - (\nabla^3_{yyy}g(x',y^*(x'))|\mathrm{d}y^*(x').\epsilon)\| & \\ \|(\nabla^3_{yyx}g(x,y^*(x))|\epsilon) - (\nabla^3_{yyx}g(x',y^*(x'))|\epsilon)\|)\} & \\ \leq C_f\left\{2\frac{2L_{g,2}(1+L^*)}{\mu_g^3} + \frac{L_{g,3}(1+L^*)}{\mu_g^2}\right\}\|x-x'\|\end{aligned}$$

Thus v^* is L_{vx}-smooth with

$$\begin{aligned}L_{vx} = \frac{L_{f,2}L_*(1+L_*) + L_{yx}L_{f,1} + L_{f,2}(1+L_*)}{\mu_g} + 2\frac{(L_{g,2}+L_{f,1})(L_*+1)}{\mu_g^2} & \\ + \frac{C_f L_{g,3}(1+L^*)}{\mu_g^2} + 4\frac{C_f L_{g,2}(1+L^*)}{\mu_g^3} \ .\end{aligned}$$

□

A.1.2 Proof of Lemma 4.2

We now provide the proof of Lemma 4.2.

Proof. **Inequality for δ_y.**

We start by expanding the square:

$$\begin{aligned}\|y^{t+1} - y^*(x^{t+1})\|^2 = \|y^{t+1} - y^*(x^t)\|^2 + \|y^*(x^{t+1}) - y^*(x^t)\|^2 & \\ - 2\langle y^{t+1} - y^*(x^t), y^*(x^{t+1}) - y^*(x^t)\rangle & \end{aligned} \tag{A.2}$$

We study each member, using the unbiasedness of D_z^t and the μ_g-strong convexity of $g(\,\cdot\,, x^t)$:

$$\begin{aligned}\mathbb{E}_t[\|y^{t+1} - y^*(x^t)\|^2] &= \mathbb{E}_t[\|y^t - y^*(x^t)\|^2] - 2\rho\mathbb{E}_t[\langle D_z^t, y^t - y^*(x^t)\rangle] + \rho^2\mathbb{E}_t[\|D_y^t\|^2] \\ &= \|y^t - y^*(x^t)\|^2 - 2\rho\langle\nabla_y g(x^t, y^t), y^t - y^*(x^t)\rangle + \rho^2\mathbb{E}_t[\|D_y^t\|^2] \\ &\leq (1 - \rho\mu_g)\|y^t - y^*(x^t)\|^2 + \rho^2\mathbb{E}_t[\|D_y^t\|^2] \ .\end{aligned}$$

Taking the total expectation yields

$$\mathbb{E}[\|y^{t+1} - y^*(x^t)\|^2] \leq (1 - \rho\mu_g)\delta_y^t + \rho^2 V_y^t \ .$$

The second member is bounded using Lipschitz continuity of y^*:

$$\mathbb{E}[\|y^*(x^{t+1}) - y^*(x^t)\|^2] \leq L_*^2\mathbb{E}[\|x^{t+1} - x^t\|^2] = L_*^2\gamma^2 V_x^t \ .$$

For the remaining scalar product, we have

$$-2\langle y^{t+1} - y^*(x^t), y^*(x^{t+1}) - y^*(x^t)\rangle = -2[\langle z^t - y^*(x^t), y^*(x^{t+1}) - y^*(x^t)\rangle - \rho\langle D_z^t, y^*(x^{t+1}) - y^*(x^t)\rangle] \ .$$

The second term can be bounded using the Cauchy-Schwarz inequality, the Lipschitz-continuity of y^*, and the Young inequality:

$$\begin{aligned}\mathbb{E}[\rho\langle D_z^t, y^*(x^{t+1}) - y^*(x^t)\rangle] &\leq \mathbb{E}[\rho\|D_z^t\|\|y^*(x^{t+1}) - y^*(x^t)\|] \\ &\leq \rho L_*\mathbb{E}[\|D_z^t\|\|x^{t+1} - x^t\|] \\ &\leq \frac{\rho^2}{2}V_y^t + \frac{L_*^2}{2}\|x^{t+1} - x^t\|^2 \\ &\leq \frac{\rho^2}{2}V_y^t + L_*^2\frac{\gamma^2}{2}V_x^t \ .\end{aligned}$$

For $-2\langle y^t - y^*(x^t), y^*(x^{t+1}) - y^*(x^t)\rangle$, we follow the proof of Chen et al. (2021) which consists in making appear the "unbiased part of $y^*(x^{t+1}) - y^*(x^t)$ by a linear approximation. More precisely, we have

$$\begin{aligned}\langle y^t - y^*(x^t), y^*(x^{t+1}) - y^*(x^t)\rangle = &\underbrace{\langle y^t - y^*(x^t), \mathrm{d}y^*(x^t)(x^{t+1} - x^t)\rangle}_{A} \\ &\underbrace{\langle y^t - y^*(x^t), y^*(x^{t+1}) - y^*(x^t) - \mathrm{d}y^*(x^t)(x^{t+1} - x^t)\rangle}_{B} \ .\end{aligned}$$

For A, we use the unbiasedness of D_x^t, Cauchy-Schwarz inequality, the Lipschitz continuity of y^* (Lemma A.1) and the identity $ab \leq \eta a^2 + \frac{b^2}{\eta}$ for any $\eta > 0$:

$$\begin{aligned}-2\mathbb{E}[A] &= -2\gamma\mathbb{E}[\langle y^t - y^*(x^t), \mathrm{d}y^*(x^t)D_x^t\rangle] \\ &= -2\gamma\mathbb{E}[\langle y^t - y^*(x^t), \mathrm{d}y^*(x^t)\mathbb{E}_t[D_x^t]\rangle] \\ &= -2\gamma\mathbb{E}[\langle y^t - y^*(x^t), \mathrm{d}y^*(x^t)D_x(y^t, v^t, x^t)\rangle] \\ &\leq 2\gamma\mathbb{E}[\|y^t - y^*(x^t)\|\|\mathrm{d}y^*(x^t)D_x(y^t, v^t, x^t)\|] \\ &\leq 2L_*\gamma\mathbb{E}[\|y^t - y^*(x^t)\|\|D_x(y^t, v^t, x^t)\|] \\ &\leq 2\eta\delta_y^t + \frac{2L_*^2}{\eta}\gamma^2\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \ .\end{aligned}$$

We take $\eta = \frac{\rho\mu_g}{4}$ and we get

$$-2\mathbb{E}[A] \leq \frac{\rho\mu_g}{2}\delta_y^t + \frac{8L_*^2}{\mu_g}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \ .$$

For B, we use Cauchy-Schwarz inequality, the smoothness of y^* (Lemma A.2), Young inequality and the boundedness of $\mathbb{E}_t[\|D_x^t\|^2]$ to get

$$\begin{aligned}
-2\mathbb{E}[B] &\leq 2\mathbb{E}[\|y^t - y^*(x^t)\|\|y^*(x^{t+1}) - y^*(x^t) - \mathrm{d}y^*(x^t)(x^{t+1} - x^t)\|] \\
&\leq L_{yx}\mathbb{E}[\|y^t - y^*(x^t)\|\|x^{t+1} - x^t\|^2] \\
&\leq L_{yx}\nu\mathbb{E}[\|y^t - y^*(x^t)\|^2\|x^{t+1} - x^t\|^2] + \frac{L_{yx}}{\nu}\mathbb{E}[\|x^{t+1} - x^t\|^2] \\
&\leq L_{yx}\nu\gamma^2\mathbb{E}[\|y^t - y^*(x^t)\|^2\mathbb{E}_t[\|D_x^t\|^2]] + \frac{L_{yx}\gamma^2}{\nu}V_x^t \\
&\leq L_{yx}B_x^2\nu\gamma^2\delta_y^t + \frac{L_{yx}\gamma^2}{\nu}V_x^t \ .
\end{aligned}$$

We take $\nu = \frac{L_{yx}}{L_*^2}$ and we get

$$-2\mathbb{E}[B] \leq \frac{L_{yx}^2 B_x^2\gamma^2}{L_*^2}\delta_y^t + L_*^2\gamma^2 V_x^t$$

Now, using $\gamma^2 \leq \frac{\rho\mu_g L_*^2}{B_x^2 L_{yx}^2}$, we end up with

$$\boxed{\delta_y^{t+1} \leq (1 - \frac{\rho\mu_g}{4})\delta_y^t + 2\rho^2 V_y^t + \beta_{yx}\gamma^2 V_x^t + \overline{\beta}_{yx}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \ ,} \tag{A.3}$$

with $\beta_{yx} = 3L_*^2$ and $\overline{\beta}_{yx} = \frac{8L_*^2}{\mu_g}$.

Inequality for δ_v. We proceed in a similar way for v:

$$\delta_v^{t+1} \leq \mathbb{E}[\|v^{t+1} - v^*(x^t)\|^2] + \mathbb{E}[\|v^*(x^{t+1}) - v^*(x^t)\|^2] - 2\mathbb{E}[\langle v^{t+1} - v^*(x^t), v^*(x^{t+1}) - v^*(x^t)\rangle] \ .$$

For the first term, we have

$$\mathbb{E}_t[\|v^{t+1} - v^*(x^t)\|^2] = \|v^t - v^*(x^t)\|^2 - 2\rho\langle D_v(y^t, v^t, x^t), v^t - v^*(x^t)\rangle + \rho^2\mathbb{E}_t[\|D_v^t\|^2]$$

Now, using that $D_v(y^*(x^t), v^*(x^t), x^t) = 0$:

$$\begin{aligned}
\langle D_v(y^t, v^t, x^t), v^t - v^*(x^t)\rangle &= \langle D_v(y^t, v^t, x^t) - D_v(y^*(x^t), v^*(x^t), x^t), v^t - v^*(x^t)\rangle \\
&= \langle \nabla_{yy}^2 g(x^t, y^t)(v^t - v^*(x^t)), v^t - v^*(x^t)\rangle \\
&\quad + \langle (\nabla_{yy}^2 g(x^t, y^t) - \nabla_{yy}^2 g(y^*(x^t), x^t))v^*(x^t), v^t - v^*(x^t)\rangle \\
&\quad + \langle (\nabla_y f(x^t, y^t) - \nabla_1 f(y^*(x^t), x^t)), v^t - v^*(x^t)\rangle \\
&\geq \mu_g\|v^t - v^*(x^t)\|^2 - \frac{L_{g,2}C_f}{\mu_g}\|y^t - y^*(x^t)\|\|v^t - v^*(x^t)\| \\
&\quad - L_{f,1}\|y^t - y^*(x^t)\|\|v^t - v^*(x^t)\| \\
&\geq \mu_g\|v^t - v^*(x^t)\|^2 - \omega\|y^t - y^*(x^t)\|\|v^t - v^*(x^t)\|
\end{aligned}$$

where $\omega = L_{f,1} + \frac{L_{g,2}C_f}{\mu_g}$. We then use $\omega\|y^t - y^*(x^t)\|\|v^t - v^*(x^t)\| \leq \frac{1}{2}c\|v^t - v^*(x^t)\|^2 + \frac{\omega^2}{2c}\|y^t - y^*(x^t)\|^2$ with $c = \mu_g$ to get

$$-\langle D_v(y^t, v^t, x^t), v^t - v^*(x^t)\rangle \leq -\frac{1}{2}\mu_g\delta_v^t + \frac{\omega^2}{2\mu_g}\delta_y^t \ .$$

We get the overall inequality by taking the total expectation

$$\mathbb{E}[\|v^{t+1} - v^*(x^t)\|^2] \leq \left(1 - \frac{\rho\mu_g}{2}\right)\delta_v^t + \rho\frac{\omega^2}{2\mu_g}\delta_y^t + \rho^2 V_v^t \ .$$

We also use Lipschitz on v^* to bound the other term

$$\mathbb{E}[\|v^*(x^{t+1}) - v^*(x^t)\|^2] \leq L_*^2\gamma^2 V_x^t \ .$$

As previously, the scalar product is bounded by:

$$\begin{aligned}-\mathbb{E}[\langle v^{t+1}-v^*(x^t), v^*(x^{t+1})-v^*(x^t)\rangle] &= -\mathbb{E}[\langle v^t-v^*(x^t), v^*(x^{t+1})-v^*(x^t)\rangle] \\ &\quad -\rho\mathbb{E}[\langle D_v^t, v^*(x^{t+1})-v^*(x^t)\rangle] \\ &\leq \mathbb{E}[\langle y^t-y^*(x^t), v^*(x^{t+1})-v^*(x^t)\rangle] + \frac{\rho^2}{2}V_v^t + L_*^2\frac{\gamma^2}{2}V_x^t\end{aligned}$$

We do similar manipulations pour v^*, thanks to Lemma A.3. We have as for z from Lemma A.1 for any $\eta > 0$:

$$-\mathbb{E}[\langle v^t-v^*(x^t), \mathrm{d}v^*(x^t)(x^{t+1}-x^t)\rangle] \leq \eta\delta_v^t + \frac{L_*^2\gamma^2}{\eta}\mathbb{E}[\|D_x(y^t,v^t,x^t)\|^2] \ .$$

We take $\eta = \frac{\rho\mu_g}{8}$ and we get

$$-\mathbb{E}[\langle v^t-v^*(x^t), \mathrm{d}v^*(x^t)(x^{t+1}-x^t)\rangle] \leq \frac{\rho\mu_g}{8}\delta_v^t + \frac{8L_*^2\gamma^2}{\mu_g\rho}\mathbb{E}[\|D_x(y^t,v^t,x^t)\|^2]$$

Then smoothness of v^* for any $\eta > 0$ gives us

$$-\mathbb{E}[\langle v^t-v^*(x^t), v^*(x^{t+1})-v^*(x^t)-\mathrm{d}v^*(x^t)(x^{t+1}-x^t)\rangle] \leq \frac{L_{vx}B_x^2\nu}{2}\gamma^2\delta_v^t + \frac{L_{vx}}{2\nu}\gamma^2V_x^t \ .$$

With $\nu = \frac{L_{vx}}{L_*^2}$ we get

$$-\mathbb{E}[\langle v^t-v^*(x^t), v^*(x^{t+1})-v^*(x^t)-\mathrm{d}v^*(x^t)(x^{t+1}-x^t)\rangle] \leq \frac{L_{vx}^2B_x^2}{2L_*^2}\gamma^2\delta_v^t + \frac{L_*^2}{2}\gamma^2V_x^t \ .$$

With the assumption $\gamma^2 \leqslant \frac{\rho\mu_g L_*^2}{8L_{vx}^2B_x^2}$, we get

$$\begin{aligned}\delta_v^{t+1} &\leq \left(1-\frac{\rho\mu_g}{2}+\frac{\rho\mu_g}{4}+\frac{L_{vx}^2B_x}{L_*^2}\right)\delta_v^t + \rho\beta_{vy}\delta_y^t + 2\rho^2V_y^t + 3L_*^2\gamma^2V_x^t + \frac{16L_*^2\gamma^2}{\mu_g\rho}\mathbb{E}[\|D_x(y^t,v^t,x^t)\|^2] \\ &\leq \left(1-\frac{\rho\mu_g}{8}\right)\delta_v^t + \rho\beta_{vy}\delta_y^t + 2\rho^2V_y^t + 3L_*^2\gamma^2V_x^t + \frac{16L_*^2\gamma^2}{\mu_g\rho}\mathbb{E}[\|D_x(y^t,v^t,x^t)\|^2] \ .\end{aligned}$$

And finally we have

$$\boxed{\delta_v^{t+1} \leq \left(1-\frac{\rho\mu_g}{8}\right)\delta_v^t + \rho\beta_{vy}\delta_y^t + 2\rho^2V_y^t + \beta_{vx}\gamma^2V_x^t + \overline{\beta}_{vx}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t,v^t,x^t)\|^2]} \tag{A.4}$$

with $\beta_{vy} = \frac{\omega^2}{2\mu_g}$, $\beta_{vx} = 3L_*^2$ and $\overline{\beta}_{vx} = \frac{16L_*^2\gamma^2}{\mu_g}$.

□

A.1.3 Proof of Theorem 4.1

This section is devoted to the proof of Theorem 4.1 that we recall here.

Theorem 4.1 (Convergence of SOBA, fixed step size)**.** *Fix an iteration $T > 1$ and assume that Assumptions 4.1 to 4.5 hold. We consider fixed steps $\rho^t = \frac{\overline{\rho}}{\sqrt{T}}$ and $\gamma^t = \xi\rho^t$ with $\overline{\rho}$ and ξ precised in the appendix. Let $(x^t)_{t\geq 1}$ the sequence of outer iterates for SOBA. Then,*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\Phi(x^t)\|^2] = \mathcal{O}(T^{-\frac{1}{2}}) \ .$$

The values of the different constants are

$$\phi'_y = \frac{1}{8\overline{\beta}_{yx}}, \quad \phi'_v = \min\left(\frac{1}{8\overline{\beta}_{vx}}, \frac{\mu_g \phi'_y}{32\beta_{vy}}\right) \quad , \quad \overline{\rho} = \min\left(\frac{16}{\mu_g}, \frac{\mu_g}{16L_y^2 B_y^2}, \frac{\mu_g}{32L_v^2 B_v^2}, \frac{\beta_{vy}}{L_v^2 B_v^2}\right) \quad ,$$
$$\text{and} \quad \xi^2 = \frac{\mu_g}{4} \min\left[\min\left(\frac{1}{L_{yx}^2}, \frac{1}{L_{vx}^2}\right) \frac{L_*^2}{B_x^2 \overline{\rho}}, \min\left(\phi'_v, \phi'_y\right) \frac{1}{2L_x^2}\right] \quad .$$

Before, one has to adapt our descent lemmas to the case of SOBA.

Lemma A.4. *Assume that the step sizes ρ and γ verify*

$$\rho \leq \min\left(\frac{\mu_g}{16L_y^2 B_y^2}, \frac{\mu_g}{32L_v^2 B_v^2}, \frac{\beta_{vy}}{L_v^2 B_v^2}\right), \quad \textit{and} \quad \gamma^2 \leq \min\left(\frac{\rho\mu_g L_*^2}{4B_x^2 L_{yx}^2}, \frac{\rho\mu_g L_*^2}{8B_x^2 L_{vx}^2}\right) \quad .$$

Then it holds

$$\delta_y^{t+1} \leq \left(1 - \frac{\rho\mu_g}{8}\right)\delta_y^t + 2\rho^2 B_y^2 + \beta_{yx}\gamma^2 B_x^2 + \overline{\beta}_{yx}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2]$$
$$\delta_v^{t+1} \leq \left(1 - \frac{\rho\mu_g}{16}\right)\delta_v^t + 2\beta_{vy}\rho\delta_y^t + 2\rho^2 B_v^2 + \beta_{vx}\gamma^2 B_x^2 + \overline{\beta}_{vx}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \quad .$$

Proof. From Assumption 4.4 and Lemma 4.1, we have

$$V_y^t \leq B_y^2(1 + D_y(y^t, v^t, x^t)) \leq B_y^2(1 + L_y^2\delta_y^t) \quad .$$

Plugging this into Equation (A.3) and using $V_x^t \leq B_x^2$ yields

$$\delta_y^{t+1} \leq \left(1 - \frac{\rho\mu_g}{4} + 2L_y^2 B_y^2\rho^2\right)\delta_y^t + 2\rho^2 B_y^2 + \beta_{yx}\gamma^2 B_x^2 + \overline{\beta}_{yx}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \quad .$$

Since by assumption $\rho \leq \frac{\mu_g}{16L_y^2 B_y^2}$, we have

$$\delta_y^{t+1} \leq \left(1 - \frac{\rho\mu_g}{8}\right)\delta_y^t + 2\rho^2 B_y^2 + \beta_{yx}\gamma^2 B_x^2 + \overline{\beta}_{yx}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \quad .$$

For δ_v^t, Assumption 4.3 and Lemma 4.1 provide us

$$V_v^t \leq B_v^2(1 + L_v^2(\delta_y^t + \delta_v^t)) \quad .$$

Since the assumptions of Lemma 4.2 are verified, we can plug the previous inequality into Equation (A.4) to get

$$\delta_v^{t+1} \leq \left(1 - \frac{\rho\mu_g}{8} + 2L_v^2 B_v^2\rho^2\right)\delta_v^t + (\beta_{vy}\rho + 2L_v^2\rho^2 B_v^2)\delta_y^t + 2\rho^2 B_v^2 + \beta_{vx}\gamma^2 B_x^2 + \overline{\beta}_{vx}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2]$$

which can be simplified using $\rho \leq \min\left(\frac{\mu_g}{32L_v^2 B_v^2}, \frac{\beta_{vy}}{L_v^2 B_v^2}\right)$ to get finally

$$\delta_v^{t+1} \leq \left(1 - \frac{\rho\mu_g}{16}\right)\delta_v^t + 2\beta_{vy}\rho\delta_y^t + 2\rho^2 B_v^2 + \beta_{vx}\gamma^2 B_x^2 + \overline{\beta}_{vx}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \quad .$$

□

We can now prove Theorem 4.1.

Proof. Consider the Lyapunov function $\mathcal{L}^t = \Phi^t + \phi_y\delta_y^t + \phi_v\delta_v^t$. Using the Equations (4.9), (A.3) and (A.4), we can bound $\mathcal{L}^{t+1} - \mathcal{L}^t$:

$$\begin{aligned}
\mathcal{L}^{t+1} - \mathcal{L}^t \leq & -\frac{\gamma}{2}g^t - \left(\frac{\gamma}{2} - \phi_y\overline{\beta}_{yx}\frac{\gamma^2}{\rho} - \phi_v\overline{\beta}_{vx}\frac{\gamma^2}{\rho}\right)\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \\
& - \left(\phi_y\frac{\mu_g}{8}\rho - \frac{L_x^2}{2}\gamma - 2\phi_v\beta_{vy}\rho\right)\delta_y^t \\
& - \left(\phi_v\frac{\mu_g}{16}\rho - \frac{L_x^2}{2}\gamma\right)\delta_v^t \\
& + \left(\frac{L_\Phi}{2} + \phi_y\beta_{yx} + \phi_v\beta_{vx}\right)B_x^2\gamma^2 \\
& + 2(\phi_y B_y^2 + \phi_v B_v^2)\rho^2 \quad .
\end{aligned}$$

Let $\phi'_y = \phi_y \frac{\gamma}{\rho}$ and $\phi'_v = \phi_v \frac{\gamma}{\rho}$, so that:

$$\begin{aligned}\mathcal{L}^{t+1} - \mathcal{L}^t \leq & -\frac{\gamma}{2} g^t - \left(\frac{\gamma}{2} - \phi'_y \overline{\beta}_{yx}\gamma - \phi'_v \overline{\beta}_{vx}\gamma\right) \mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \\ & - \left(\phi'_y \frac{\mu_g}{8}\frac{\rho^2}{\gamma} - \frac{L_x^2}{2}\gamma - 2\phi'_v \beta_{vy}\frac{\rho^2}{\gamma}\right)\delta_y^t \\ & - \left(\phi'_v \frac{\mu_g}{16}\frac{\rho^2}{\gamma} - \frac{L_x^2}{2}\gamma\right)\delta_v^t \\ & + \left(\frac{L_\Phi}{2} + \phi'_y \beta_{yx}\frac{\rho}{\gamma} + \phi_v \beta_{vx}\frac{\rho}{\gamma}\right) B_x^2 \gamma^2 \\ & + 2\left(\phi'_y B_y^2 \frac{\rho}{\gamma} + \phi'_v B_v^2 \frac{\rho}{\gamma}\right)\rho^2 \ .\end{aligned}$$

In order to get a decrease, ϕ'_y, ϕ'_v, ρ and γ must verify

$$\begin{cases} \phi'_y \overline{\beta}_{yx} + \phi'_v \overline{\beta}_{vx} \leq \frac{1}{2} \\ \frac{L_x^2}{2}\gamma + 2\phi'_v \beta_{vy}\frac{\rho^2}{\gamma} \leq \phi'_y \frac{\mu_g}{8}\frac{\rho^2}{\gamma} \\ \frac{L_x^2}{2}\gamma \leq \phi'_v \frac{\mu_g}{16}\frac{\rho^2}{\gamma} \end{cases} \tag{A.5}$$

Let us take $\phi'_y = \frac{1}{8\overline{\beta}_{yx}}$ and $\phi'_v = \min\left(\frac{1}{8\overline{\beta}_{vx}}, \frac{\mu_g \phi'_y}{32\beta_{vy}}\right)$. We have

$$\phi'_y \overline{\beta}_{yx} + \phi'_v \overline{\beta}_{vx} \leq \frac{1}{4} < \frac{1}{2}$$

and

$$\frac{L_x^2}{2}\gamma + 2\phi'_v \beta_{vy}\frac{\rho^2}{\gamma} \leq \frac{L_x^2}{2}\gamma + \phi'_y \frac{\mu_g}{16}\frac{\rho^2}{\gamma} \ .$$

If we impose $\frac{L_x^2}{2}\gamma + \phi'_y \frac{\mu_g}{16}\frac{\rho^2}{\gamma} \leq \phi'_y \frac{\mu_g}{8}\frac{\rho^2}{\gamma}$, this combined with the third condition in Equation (A.5) gives the condition $\frac{L_x^2}{2}\gamma^2 \leq \min\left(\phi'_v, \phi'_y\right)\frac{\mu_g}{16}\rho^2$. We also have the conditions coming from the assumptions of A.4, that is

$$\rho \leq \overline{\rho} = \min\left(\frac{16}{\mu_g}, \frac{\mu_g}{16 L_y^2 B_y^2}, \frac{\mu_g}{32 L_v^2 B_v^2}, \frac{\beta_{vy}}{L_v^2}\right) \tag{A.6}$$

and $\gamma^2 \leq \min\left(\frac{1}{L_{yx}^2}, \frac{1}{L_{vx}^2}\right)\frac{\mu_g L_*^2}{4B_x^2}\rho$. Let us take $\rho = \frac{\overline{\rho}}{\sqrt{T}}$ with $\gamma = \xi\rho$ where ξ is defined as

$$\xi^2 \triangleq \frac{\mu_g}{4}\min\left[\min\left(\frac{1}{L_{yx}^2}, \frac{1}{L_{vx}^2}\right)\frac{L_*^2}{B_x^2 \overline{\rho}}, \min\left(\phi'_v, \phi'_y\right)\frac{1}{2L_x^2}\right] \ . \tag{A.7}$$

From now, we have

$$\mathcal{L}^{t+1} - \mathcal{L}^t \leq -\frac{\gamma}{2} g^t + \frac{L_\Phi}{2} B_x^2 \gamma^2 + \left(\phi'_y \beta_{yx} + \phi'_v \beta_{vx}\right) B_x^2 \rho\gamma + 2\left(\phi'_y B_y^2 + \phi'_v B_v^2\right)\frac{\rho^3}{\gamma} \ . \tag{A.8}$$

Summing and telescoping yields

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^T g^t &\leq \frac{2\mathcal{L}^1}{T\gamma} + L_\Phi B_x^2 \gamma + 2\left(\phi'_y \beta_{yx} + 2\phi'_v \beta_{vx}\right) B_x^2 \rho + 4\left(\phi'_y B_y^2 + \phi'_v B_v^2\right)\frac{\rho^3}{\gamma^2} \\ &\leq \frac{2\mathcal{L}^1}{\sqrt{T}\xi\overline{\rho}} + L_\Phi B_x^2 \frac{\xi\alpha}{\sqrt{T}} + \left(\phi'_y \beta_{yx} + 2\phi'_v \beta_{vx}\right) B_x^2 \frac{\alpha}{\sqrt{T}} + 4\left(\phi'_y B_y^2 + \phi'_v B_v^2\right)\frac{\alpha}{\xi^2\sqrt{T}}\end{aligned}$$

and so

$$\boxed{\frac{1}{T}\sum_{t=1}^T g^t = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) \ .}$$

□

A.1.4 Proof of Theorem 4.2

Proof. In the decreasing step size case, we take $\rho^t = \overline{\rho}\sqrt{t}$ and $\gamma^t = \xi\rho^t$ where $\overline{\rho}$ is defined in Equation (A.6) and ξ is defined in Equation (A.7). We recall the integral majorization:

$$\sum_{t=1}^{T} t^{-1} \leq 1 + \int_1^T t^{-1}\mathrm{d}t = 1 + \log(T) \ .$$

With such definition of ρ^t and γ^t, Equation (A.8) is still valid for any $t \geqslant 1$. The only difference is that the step sizes decrease with t. Hence, by summing and rearranging in Equation (A.8), we get

$$\sum_{t=1}^{T} \gamma^t g^t \leq 2\mathcal{L}^1 + \left(L_\Phi + 2\left(\phi'_y\beta_{yx} + 2\phi'_v\beta_{vx}\right) B_x^2 \frac{1}{\xi} + 4\left(\phi'_y B_y^2 + \phi'_v B_v^2\right)\frac{1}{\xi^3}\right) \sum_{t=1}^{T} (\gamma^t)^2 \tag{A.9}$$

The left-hand-side in Equation (A.9) can be lower bounded by

$$\sum_{t=1}^{T} \gamma^t g^t \geq \left(\inf_{t\in[T]} g^t\right) \xi\overline{\rho} \sum_{t=1}^{T} t^{-\frac{1}{2}} \geq \left(\inf_{t\in[T]} g^t\right) \xi\overline{\rho} T^{\frac{1}{2}} \ . \tag{A.10}$$

Also we have

$$\sum_{t=1}^{T} (\gamma^t)^2 = \xi^2\overline{\rho}^2 \sum_{t=1}^{T} t^{-1} \leq \xi^2\overline{\rho}^2(1 + \log(T)) \ . \tag{A.11}$$

Plugging Equations (A.10) and (A.11) into Equation (A.9) and rearranging give

$$\inf_{t\in[T]} g^t \leq \frac{2\mathcal{L}^1}{\xi\overline{\rho}\sqrt{T}} + \xi\overline{\rho}\left(L_\Phi + 2\left(\phi'_y\beta_{yx} + 2\phi'_v\beta_{vx}\right) B_x^2 \frac{1}{\xi} + 4\left(\phi'_y B_y^2 + \phi'_v B_v^2\right)\frac{1}{\xi^3}\right) \frac{1 + \log(T)}{\sqrt{T}}$$

that is to say

$$\boxed{\inf_{t\in[T]} g^t = \mathcal{O}\left(\frac{1}{\sqrt{T}} + \frac{\log(T)}{\sqrt{T}}\right) \ .}$$

□

A.1.5 Proof of Theorem 4.3

In this section, we prove Theorem 4.3 that we recall here

Theorem 4.3 (Convergence of SABA, smooth case)**.** *Assume that Assumptions 4.1 to 4.3 and 4.5 to 4.6 hold. We suppose $\rho = \rho' N^{-\frac{2}{3}}$ and $\gamma = \xi\rho$, where ρ' and ξ depend only on f and g and are specified in appendix. Let x^t the iterates of SABA. Then,*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|\nabla\Phi(x^t)\|^2] = O\left(N^{\frac{2}{3}}T^{-1}\right) \ .$$

The constants ρ' and ξ are given by

$$\rho' = \min\left(\sqrt{\frac{K_1}{K_5}}, \left(\frac{K_2}{K_5}\right)^{\frac{2}{5}}, \left(\frac{K_3}{K_5}\right)^{\frac{5}{7}}, \left(\frac{K_4}{K_5}\right)^{\frac{1}{3}}, \frac{\mu_g}{64L_y^2}, \frac{\overline{\beta}_{yx}}{2\beta_{yx}}, \frac{\mu_g}{128(L_v^2 + L_v'')}, \frac{\beta_{vy}}{8(L_v^2 + L_v'')}, \frac{\overline{\beta}_{vx}}{2\beta_{vx}}\right)$$

and

$$\xi = \min(K_1, K_2(\rho')^{-\frac{1}{2}}, K_3(\rho')^{-\frac{3}{2}}, K_4(\rho')^{-1})$$

where

$$\phi''_y = \frac{1}{32\overline{\beta}_{yx}}, \quad \phi''_v = \min\left(\frac{1}{32\overline{\beta}_{vx}}, \phi''_y\frac{\mu_g}{128\beta_{vy}}\right) \ ,$$

$$K_1 = \min\left(\sqrt{\frac{\phi_y''\mu_g}{32L_x^2}}, \sqrt{\frac{\phi_v''\mu_g}{48L_x^2}}, \sqrt{\frac{L_y'}{2L_x'\beta_{yx}}}, \sqrt{\frac{L_v'}{2L_x'\beta_{vx}}}\right) \ ,$$

$$K_2 = \min\left(\sqrt{\frac{\mu_g}{64\beta_{yx}L_x''}}, \sqrt{\frac{\mu_g}{128\beta_{vx}L_x''}}, \sqrt{\frac{\beta_{vy}}{4L_x''\beta_{vx}}}\right) \ ,$$

$$K_3 = \sqrt{\frac{\phi_v''\mu_g}{384\phi_y''L_x''}} \ , \quad K_4 = \min\left(\frac{1}{4L_\Phi}, \frac{L_x^2}{2L_\Phi L_x''}, \sqrt{\frac{\Gamma'}{6L_\Phi L_x'}}, \frac{1}{8P'}, \frac{\phi_y''\mu_g}{32\beta_{sz}'}, \frac{\phi_v''\mu_g}{48\beta_{sv}'}\right)$$

and

$$K_5 = \frac{15(\phi_y''L_y' + \phi_v''L_v')}{\Gamma'} \ .$$

Control of distance from memory to iterates

We can view our method has having two "parallel" memories for each variable (y_i^t, v_i^t, x_i^t) for $i \in [n]$ corresponding to calls in g and $(y_j'^t, v_j'^t, x_j'^t)$ for $j \in [m]$ corresponding to calls to f. At each iteration, we sample i at random uniformly and do $(y_i^{t+1}, v_i^{t+1}, x_i^{t+1}) = (y^t, v^t, x^t)$ and $(y_{i'}^{t+1}, v_{i'}^{t+1}, x_{i'}^{t+1}) = (y_{i'}^t, v_{i'}^t, x_{i'}^t)$ for $i' \neq i$, and similarly for the other memory.

In what follows, we focus on controlling the error between the iterates and the memories. We define to make things simpler

$$E_y^t = \frac{1}{n}\sum_{i=1}^n \mathbb{E}[\|y^t - y_i^t\|^2] \ , \quad E_v^t = \frac{1}{n}\sum_{i=1}^n \mathbb{E}[\|v^t - v_i^t\|^2] \ , \quad E_x^t = \frac{1}{n}\sum_{i=1}^n \mathbb{E}[\|x^t - x_i^t\|^2] \ ,$$

and similarly $E_x'^t$, $E_v'^t$ and $E_x'^t$.

Lemma A.5. *We have the following inequalities:*

$$E_y^{t+1} \leq \left(1 - \frac{1}{2n}\right) E_y^t + \rho^2\mathbb{E}[\|D_y^t\|^2] + 2n\rho^2\mathbb{E}[\|D_y(y^t, v^t, x^t)\|^2] \ ,$$

$$E_v^{t+1} \leq \left(1 - \frac{1}{2n}\right) E_v^t + \rho^2\mathbb{E}[\|D_v^t\|^2] + 2n\rho^2\mathbb{E}[\|D_v(y^t, v^t, x^t)\|^2] \ ,$$

$$E_x^{t+1} \leq \left(1 - \frac{1}{2n}\right) E_x^t + \gamma^2\mathbb{E}[\|D_x^t\|^2] + 2n\gamma^2\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \ ,$$

$$E_y'^{t+1} \leq \left(1 - \frac{1}{2m}\right) E_y^t + \rho^2\mathbb{E}[\|D_y^t\|^2] + 2m\rho^2\mathbb{E}[\|D_y(y^t, v^t, x^t)\|^2] \ ,$$

$$E_v'^{t+1} \leq \left(1 - \frac{1}{2m}\right) E_v^t + \rho^2\mathbb{E}[\|D_v^t\|^2] + 2m\rho^2\mathbb{E}[\|D_v(y^t, v^t, x^t)\|^2] \ ,$$

and

$$E_x'^{t+1} \leq \left(1 - \frac{1}{2m}\right) E_x^t + \gamma^2\mathbb{E}[\|D_x^t\|^2] + 2m\gamma^2\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \ .$$

Proof. We provide the detailed proof for E_y^t. The approach for the five others is similar.

Let $i \in [n]$. Taking the expectation of $\|y^{t+1} - y_i^{t+1}\|^2$ conditionaly to y^t, v^t, x^t yields

$$\mathbb{E}_t[\|y^{t+1} - y_i^{t+1}\|^2] = \frac{1}{n}\mathbb{E}_t[\|y^{t+1} - y^t\|^2] + \frac{n-1}{n}\mathbb{E}_t[\|y^{t+1} - y_i^t\|^2] \ .$$

Then, using the fact that $\mathbb{E}_t[D_y^t] = D_y(y^t, v^t, x^t)$, we have

$$\mathbb{E}_t[\|y^{t+1} - y_i^t\|^2] = \mathbb{E}_t[\|y^{t+1} - y^t\|^2] + \|y^t - y_i^t\|^2 - 2\rho\langle D_y(y^t, v^t, x^t), y^t - y_i^t\rangle \ .$$

We then upper-bound crudely the scalar product by Cauchy-Schwarz and Young inequalities with parameter β:

$$\mathbb{E}_t[\|y^{t+1} - y_i^t\|^2] \leq \mathbb{E}_t[\|y^{t+1} - y^t\|^2] + \rho\beta^{-1}\|D_y(y^t, v^t, x^t)\|^2 + (1+\rho\beta)\|y^t - y_i^t\|^2$$

As a consequence, by taking the total expectation and summing for all $i \in [n]$, we find

$$E_y^{t+1} \leq \rho^2\mathbb{E}[\|D_y^t\|^2] + \rho\beta^{-1}\left(1 - \frac{1}{n}\right)\mathbb{E}[\|D_y(y^t, v^t, x^t)\|^2] + (1+\rho\beta)\left(1 - \frac{1}{n}\right)E_y^t \ .$$

Finally, we take $\beta = \frac{1}{2n\rho}$ to obtain

$$\boxed{E_y^{t+1} \leq \left(1 - \frac{1}{2n}\right)E_y^t + \rho^2\mathbb{E}[\|D_y^t\|^2] + 2n\rho^2\mathbb{E}[\|D_y(y^t, v^t, x^t)\|^2] \ .} \tag{A.12}$$

□

Bounds on the variances

The following lemma gives us upper-bounds for $\mathbb{E}[\|D_y^t\|^2]$, $\mathbb{E}[\|D_v^t\|^2]$, and $\mathbb{E}[\|D_x^t\|^2]$.

Lemma A.6. *For SABA, there are constants $L'_y, L'_v, L'_x > 0$ such that*

$$\mathbb{E}[\|D_y^t\|]^2 \leq 2\mathbb{E}[\|D_y(y^t, v^t, x^t)\|^2] + 2L'_y(E_y^t + E_x^t) \ ,$$

$$\mathbb{E}[\|D_v^t\|^2] \leq 2\mathbb{E}[\|D_v(y^t, v^t, x^t)\|^2] + 2L'_v(E_y^t + E_x^t + E_v^t + E_y'^t + E_x'^t) + 2L''_v(\delta_y^t + \delta_v^t)$$

and

$$\mathbb{E}[\|D_x^t\|^2] \leq 2\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] + 2L'_x(E_y^t + E_x^t + E_v^t + E_y'^t + E_x'^t) + 2L''_x(\delta_y^t + \delta_v^t) \ .$$

Proof. For SABA, if we consider i sampled from $[n]$ at iteration t, we have

$$D_y^t = \nabla_y g_i(y^t, x^t) - \nabla_y g_i(y_i^t, x_i^t) + \frac{1}{n}\sum_{i'=1}^n \nabla_y g_{i'}(y_{i'}^t, x_{i'}^t) \ .$$

Hence we get

$$\begin{aligned}\mathbb{E}_t[\|D_y^t\|^2] &= \mathbb{E}_t[\|\nabla_y g_i(y^t, x^t) - \nabla_y g_i(y_i^t, x_i^t) + \frac{1}{n}\sum_{i'=1}^N \nabla_y g_{i'}(y_{i'}^t, x_{i'}^t) \\ &\quad - \nabla_y g(x^t, y^t) + \nabla_y g(x^t, y^t)\|^2] \\ &\leq 2\|\nabla_y g(x^t, y^t)\|^2 + 2\mathbb{E}_t[\|\nabla_y g_i(y^t, x^t) - \nabla_y g_i(y_i^t, x_i^t) \\ &\quad + \frac{1}{n}\sum_{i'=1}^N \nabla_y g_{i'}(y_{i'}^t, x_{i'}^t) - \nabla_y g(x^t, y^t)\|^2] \ .\end{aligned} \tag{A.13}$$

The second term is the variance of $\nabla_y g_i(y^t, x^t) - \nabla_y g_i(y_i^t, x_i^t)$, which is therefore upper-bounded by

$$\begin{aligned}\mathbb{E}_t[\|[\nabla_y g_i(y^t, x^t) - \nabla_y g_i(y_i^t, x_i^t)\|^2] &= \frac{1}{n}\sum_{i=1}^n \|[\nabla_y g_i(y^t, x^t) - \nabla_y g_i(y_i^t, x_i^t)\|^2 \\ &\leq \frac{L'_y}{n}\sum_{i=1}^n (\|y^t - y_i^t\|^2 + \|x^t - x_i^t\|^2)\end{aligned} \tag{A.14}$$

where the inequality comes from the Lipschitz continuity of each $\nabla_y g_i$ with $L'_z = \max_{i\in[n]} L_{g_i,1}$.

Then, by plugging (A.14) into (A.13) and taking the total expectation, we get

$$\boxed{\mathbb{E}[\|D_y^t\|]^2 \leq 2\mathbb{E}[\|D_y(y^t, v^t, x^t)\|^2] + 2L'_y(E_y^t + E_x^t) \ .} \tag{A.15}$$

Things are quite similar for the other variables, albeit a bit more difficult.

In v, it holds

$$\begin{aligned}
\mathbb{E}_t[\|D_v^t\|^2] =& \mathbb{E}_t[\|\nabla_y f_j(y^t, x^t) - \nabla_y f_j(y_j'^t, x_j'^t) + \frac{1}{m}\sum_{j'=1}^{m} \nabla_y f_{j'}(y_{j'}'^t, x_{j'}'^t) \\
&+ \nabla_{yy}^2 g_i(y^t, x^t)v^t - \nabla_{yy}^2 g_i(y_i^t, x_i^t)v_i^t + \frac{1}{n}\sum_{i'=1}^{n} \nabla_y^2 g_{i'}(y_{i'}^t, x_{i'}^t)v_{i'^t} \\
&- D_v(y^t, v^t, x^t) + D_v(y^t, v^t, x^t)\|^2] \\
\leq& 2[\|D_v(y^t, v^t, x^t)\|^2 \\
&+ 2\mathbb{E}_t[\|\nabla_y f_j(y^t, x^t) - \nabla_y f_j(y_j'^t, x_j'^t) + \frac{1}{m}\sum_{j'=1}^{m} \nabla_y f_{j'}(y_{j'}'^t, x_{j'}'^t) \\
&+ \nabla_{yy}^2 g_i(y^t, x^t)v^t - \nabla_{yy}^2 g_i(y_i^t, x_i^t)v_i^t + \frac{1}{n}\sum_{i'=1}^{n} \nabla_y g_{i'}^2(y_{i'}^t, x_{i'}^t)v_{i'}^t \\
&- D_v(y^t, v^t, x^t)\|^2]
\end{aligned}$$

Here, we see that we need to control the variance of

$$\nabla_y f_j(y^t, x^t) - \nabla_y f_j(y_j'^t, x_j'^t) + \nabla_{yy}^2 g_i(y^t, x^t)v^t - \nabla_{yy}^2 g_i(y_i^t, x_i^t)v_i^t \ .$$

Since i and j are independent, this is a sum of two independent random variables, hence its variance is the sum of the variances, which is upper-bounded by

$$\mathbb{E}_t[\|\nabla_y f_j(y^t, x^t) - \nabla_y f_j(y_j'^t, x_j'^t)\|^2] + \mathbb{E}_t[\|\nabla_{yy}^2 g_i(y^t, x^t)v^t - \nabla_{yy}^2 g_i(y_i^t, x_i^t)v_i^t\|^2] \ .$$

For $\mathbb{E}_t[\|\nabla_y f_j(y^t, x^t) - \nabla_y f_j(y_j'^t, x_j'^t)\|^2]$ we use the lipschitz continuity of the $\nabla_y f_j$:

$$\begin{aligned}
\mathbb{E}_t[\|\nabla_y f_j(y^t, x^t) - \nabla_y f_j(y_j'^t, x_j'^t)\|^2] &\leq \left[\max_{j\in[m]} L_{f_j,1}\right] \mathbb{E}_t[\|y^t - z_j^t\|^2 + \|x^t - x_j^t\|^2] \\
&\leq \left[\max_{j\in[m]} L_{f_j,1}\right] \frac{1}{m}\sum_{j=1}^{m}(\|y^t - z_j^t\|^2 + \|x^t - x_j^t\|^2) \ .
\end{aligned}$$

The control of $\mathbb{E}_t[\|\nabla_{yy}^2 g_i(y^t, x^t)v^t - \nabla_{yy}^2 g_i(y_i^t, x_i^t)v_i^t\|^2]$ is a bit harder without assuming the boundness of v beforehand. But, we can bypass the difficulty by introducing $\nabla_{yy}^2 g_i(y^*(x^t), x^t)v^*(x^t)$:

$$\begin{aligned}
\mathbb{E}_t[\|\nabla_{yy}^2 g_i(y^t, x^t)v^t - \nabla_{yy}^2 g_i(y_i^t, x_i^t)v_i^t\|^2] \leq& 4\big\{\mathbb{E}_t[\|\nabla_{yy}^2 g_i(y^t, x^t)(v^t - v^*(x^t))\|^2] \\
&+ \mathbb{E}_t[\|(\nabla_{yy}^2 g_i(y^t, x^t) - \nabla_{yy}^2 g_i(y^*(x^t), x^t))v^*(x^t)\|^2] \\
&+ \mathbb{E}_t[\|(\nabla_{yy}^2 g_i(y^*(x^t), x^t) - \nabla_{yy}^2 g_i(y_i^t, x_i^t))v^*(x^t)\|^2] \\
&+ \mathbb{E}_t[\|\nabla_{yy}^2 g_i(y_i^t, x_i^t)(v^*(x^t) - v_i^t)\|^2]\big\} \\
\leq& 4(\max_{i\in[n]} L_{g_i,1})\|v^t - v^*(x^t)\|^2 + (\max_{i\in[n]} L_{g_i,2})\frac{C_f}{\mu_g}\|y^t - y^*(x^t)\|^2 \\
&+ (\max_{i\in[n]} L_{g_i,2})\frac{C_f}{\mu_g}(\|x^t - x_i^t\|^2 + 2(\|y^t - y^*(x^t)\|^2 + \|y^t - y_i^t\|^2)) \\
&+ (\max_{i\in[n]} L_{g_i,1})(\|x^t - x_i^t\|^2 + 2(\|v^t - v^*(x^t)\|^2 + \|v^t - v_i^t\|^2))
\end{aligned}$$

Let

$$L_v' = 4\max\left(2\max_{i\in[n]} L_{g_i,1}, 2\max_{i\in[n]} L_{g_i,2}\frac{C_f}{\mu_g}, \max_{j\in[m]} L_{f_j,1}\right)$$

and

$$L_v'' = 4\max\left(3\max_{i\in[n]} L_{g_i,1}), 3\max_{i\in[n]} L_{g_i,2})\frac{C_f}{\mu_g}\right) \ .$$

Taking the total expectation and putting all together yields

$$\boxed{\mathbb{E}[\|D_v^t\|^2] \leq 2\mathbb{E}[\|D_v(y^t,v^t,x^t)\|^2] + 2L'_v(E_y^t + E_x^t + E_v^t + E_y'^t + E_x'^t) + 2L''_v(\delta_y^t + \delta_v^t)} \ . \tag{A.16}$$

In x we have similarly

$$\boxed{\mathbb{E}[\|D_x^t\|^2] \leq 2\mathbb{E}[\|D_x(y^t,v^t,x^t)\|^2] + 2L'_x(E_y^t + E_x^t + E_v^t + E_y'^t + E_x'^t) + 2L''_x(\delta_y^t + \delta_v^t)} \ . \tag{A.17}$$

□

We now form $S^t = E_y^t + E_x^t + E_v^t + E_y'^t + E_v'^t + E_x'^t$, and letting $\Gamma = \min(\frac{1}{m}, \frac{1}{n})$. Note that by definition, each quantity E_y^t is smaller than S^t.

We will therefore use the cruder bounds on $\mathbb{E}[\|D_y^t\|^2]$, $\mathbb{E}[\|D_v^t\|^2]$ and $\mathbb{E}[\|D_x^t\|^2]$ as follows thanks to Lemma 4.1 and Lemma A.6

$$\mathbb{E}[\|D_y^t\|^2] \leq 2L_y^2\delta_y^t + 2L'_y S^t \ , \tag{A.18}$$

$$\mathbb{E}[\|D_v^t\|^2] \leq 2(L_v^2 + L''_v)(\delta_y^t + \delta_v^t) + 2L'_v S^t \tag{A.19}$$

and

$$\mathbb{E}[\|D_x^t\|^2] \leq 2\mathbb{E}[\|D_x\|^2] + 2L'_x S^t + 2L''_x(\delta_y^t + \delta_v^t) \ . \tag{A.20}$$

We have the following lemma

Lemma A.7. *If* $4\rho^2(L'_y + L'_v) + 4\gamma^2 L'_x \leq \frac{\Gamma}{2}$ *and* $4L''_x\gamma^2 \leq \rho^2(L_v^2 + 4L''_v)$*, it holds*

$$S^{t+1} \leq \left(1 - \frac{\Gamma}{2}\right) S^t + \beta_{sy}\rho^2\delta_y^t + \beta_{sv}\rho^2\delta_v^t + P\gamma^2\mathbb{E}[\|D_x\|^2]$$

for some $L_s, \beta_{sy}, P > 0$.

Proof. It holds following eq. (A.12) (and omitting the dependencies in (y^t, v^t, x^t) in the direction for simplicity)

$$\begin{aligned} S^{t+1} \leq (1-\Gamma)\, S^t + \mathbb{E}\left[2\rho^2(\|D_y^t\|^2 + \|D_v^t\|^2) + 2\gamma^2\|D_x^t\|^2\right. \\ \left. +2(m+n)[\rho^2(\|D_y\|^2 + \|D_v\|^2) + \gamma^2\|D_x\|^2]\right] \ . \end{aligned}$$

Using the previous bounds (A.15), (A.16) and (A.17), we get

$$\begin{aligned} S^{t+1} \leq \left(1 - \Gamma + 4\rho^2(L'_y + L'_v) + 4\gamma^2 L'_x\right) S^t + (2(m+n)+4)\mathbb{E}[\rho^2(\|D_y\|^2 + \|D_v\|^2) \\ + \gamma^2\|D_x\|^2] + 4L''_v\rho^2(\delta_y^t + \delta_v^t) + 4L''_x\gamma^2(\delta_y^t + \delta_v^t) \ . \end{aligned}$$

Next, using $4\rho^2(L'_y + L'_v) + 4\gamma^2 L'_x \leq \frac{\Gamma}{2}$ and letting $P = (2(m+n)+4)$ we get

$$S^{t+1} \leq \left(1 - \frac{\Gamma}{2}\right) S^t + P\mathbb{E}[\rho^2(\|D_y\|^2 + \|D_v\|^2) + \gamma^2\|D_x\|^2] + +4L''_v\rho^2(\delta_y^t + \delta_v^t) + 4L''_x\gamma^2(\delta_y^t + \delta_v^t) \ .$$

To finish, we use Lemma 4.1 to get

$$S^{t+1} \leq \left(1 - \frac{\Gamma}{2}\right) S^t + P[\rho^2((L_y^2 + L_v^2)\delta_y^t + L_v^2\delta_v^t) + (4L''_v\rho^2 + 4L''_x\gamma^2)(\delta_y^t + \delta_v^t) + \gamma^2\mathbb{E}[\|D_x\|^2]] \ .$$

Then, using that $4L''_x\gamma^2 \leq \rho^2(L_v^2 + 4L''_v)$, we get the bound, letting $L_{sy} = L_y^2 + L_v^2 + 4L''_v$ and $L_{sv} = L_v^2 + 4L''_v$:

$$\boxed{S^{t+1} \leq \left(1 - \frac{\Gamma}{2}\right) S^t + \beta_{sy}\rho^2\delta_y^t + \beta_{sv}\rho^2\delta_v^t + P\gamma^2\mathbb{E}[\|D_x\|^2]}$$

with $\beta_{sy} = 2PL_{sy}$, $\beta_{sv} = 2PL_{sv}$ □

Putting it all together

Recall that we denote $g^t = \mathbb{E}[\|\nabla\Phi(x^t)\|^2]$ and $\Phi^t = \mathbb{E}[\Phi(x^t)]$. In the following lemma, we adapt Lemma 4.2 and Lemma 4.3 to the SABA algorithm.

Lemma A.8. *If*

$$\rho \leq \min\left(\frac{\mu_g}{64L_y^2}, \frac{\overline{\beta}_{yx}}{2\beta_{yx}}, \frac{\mu_g}{128(L_v^2+L_v'')}, \frac{\beta_{vy}}{8(L_v^2+L_v'')}, \frac{\overline{\beta}_{vx}}{2\beta_{vx}}\right)$$

and

$$\gamma \leq \min\left(\sqrt{\frac{\rho\mu_g}{64\beta_{yx}L_x''}}, \sqrt{\frac{L_y'}{2L_x'\beta_{yx}}}\rho, \sqrt{\frac{\rho\mu_g}{128\beta_{vx}L_x''}}, \sqrt{\frac{\rho\beta_{vy}}{4L_x''\beta_{vx}}}, \sqrt{\frac{L_v'}{2L_x'\beta_{vx}}}\rho, \frac{1}{4L_\Phi}, \frac{L_x^2}{2L_\Phi L_x''}\right)$$

then it holds

$$\delta_y^{t+1} \leq \left(1-\frac{\rho\mu_g}{8}\right)\delta_y^t + 2L_x''\beta_{yx}\gamma^2\delta_v^t + 5L_y'\rho^2 S^t + 2\overline{\beta}_{yx}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t,v^t,x^t)\|^2] \ ,$$

$$\delta_v^{t+1} \leq \left(1-\frac{\rho\mu_g}{16}\right)\delta_v^t + 3\beta_{vy}\rho\delta_y^t + 5L_v'\rho^2 S^t + 2\overline{\beta}_{vx}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t,v^t,x^t)\|^2]$$

and

$$\Phi^{t+1} \leq \Phi^t - \frac{\gamma}{2}g^t - \frac{\gamma}{4}\mathbb{E}[\|D_x(y^t,v^t,x^t)\|^2] + L_x^2\gamma(\delta_y^t+\delta_v^t) + L_\Phi L_x'\gamma^2 S^t \ . \tag{A.21}$$

Proof. We start from Lemma 4.2 and plug the bounds of Equations (A.18) and (A.19).

$$\begin{aligned}\delta_y^{t+1} \leq& \left(1-\frac{\rho\mu_g}{4}+4L_y^2\rho^2+4\beta_{yx}L_x''\gamma^2\right)\delta_y^t + 2L_x''\beta_{yx}\gamma^2\delta_v^t \\ &+ (4L_y'\rho^2+2L_x'\beta_{yx}\gamma^2)S^t + \left(2\beta_{yx}\gamma^2+\overline{\beta}_{yx}\frac{\gamma^2}{\rho}\right)\mathbb{E}[\|D_x(y^t,v^t,x^t)\|^2]\end{aligned} \tag{A.22}$$

Since $\rho \leq \frac{\mu_g}{64L_y^2}$ and $\gamma^2 \leq \frac{\rho\mu_g}{64\beta_{yx}L_x''}$, we have

$$-\frac{\rho\mu_g}{4}+4L_y^2\rho^2+4\beta_{yx}L_x''\gamma^2 \leq -\frac{\rho\mu_g}{8} \ . \tag{A.23}$$

The condition $\gamma^2 \leq \frac{L_y'}{2L_x'\beta_{yx}}\rho^2$ gives us

$$4L_y'\rho^2 + 2L_x'\beta_{yx}\gamma^2 \leq 5L_y'\rho^2 \ . \tag{A.24}$$

With $\rho \leq \frac{\overline{\beta}_{yx}}{2\beta_{yx}}$, we get

$$2\beta_{yx}\gamma^2 + \overline{\beta}_{yx}\frac{\gamma^2}{\rho} \leq 2\overline{\beta}_{yx}\frac{\gamma^2}{\rho} \ . \tag{A.25}$$

We can plug Equations (A.23), (A.24) and (A.25) into Equation (A.22) and we end up with

$$\delta_y^{t+1} \leq \left(1-\frac{\rho\mu_g}{8}\right)\delta_y^t + 2L_x''\beta_{yx}\gamma^2\delta_v^t + 5L_y'\rho^2 S^t + 2\overline{\beta}_{yx}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t,v^t,x^t)\|^2] \ .$$

The proof for δ_v^t is quite similar. From Lemma 4.2, Equations (A.19) and (A.20).

$$\begin{aligned}\delta_v^{t+1} &\leq \left(1-\frac{\rho\mu_g}{8}\right)\delta_v^t + \beta_{vy}\rho\delta_y^t + 2\rho^2 V_v^t + \beta_{vx}\gamma^2 V_x^t + \overline{\beta}_{vx}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t,v^t,x^t)\|^2] \\ &\leq \left(1-\frac{\rho\mu_g}{8}+4(L_v^2+L_v'')\rho^2+4L_x''\beta_{vx}\gamma^2\right)\delta_v^t + (4(L_v^2+L_v'')\rho^2+2L_x''\beta_{vx}\gamma^2+\beta_{vy}\rho)\delta_y^t + \\ &\quad + \left(4L_v'\rho^2+2L_x'\beta_{vx}\gamma^2\right)S^t + \left(2\beta_{vx}\gamma^2+\overline{\beta}_{vx}\frac{\gamma^2}{\rho}\right)\mathbb{E}[\|D_x(y^t,v^t,x^t)\|^2] \ .\end{aligned} \tag{A.26}$$

Using $\rho \leq \frac{\mu_g}{128(L_v^2+L_v'')}$ and $\gamma^2 \leq \frac{\rho\mu_g}{128L_x''\beta_{vx}}$, we get

$$-\frac{\rho\mu_g}{8} + 4(L_v^2 + L_v'')\rho^2 + 4L_x''\beta_{vx}\gamma^2 \leq -\frac{\rho\mu_g}{16} \ . \tag{A.27}$$

With $\gamma^2 \leq \frac{\rho\beta_{vy}}{4L_x''\beta_{vx}}$ and $\rho \leq \frac{\beta_{vy}}{8(L_v^2+L_v'')}$, we have

$$4(L_v^2 + L_v'')\rho^2 + 2L_x''\beta_{vx}\gamma^2 + \beta_{vy}\rho \leq 3\beta_{vy}\rho \ . \tag{A.28}$$

The condition $\gamma^2 \leq \frac{L_v'}{2L_x'\beta_{vx}}\rho^2$ yields

$$4L_v'\rho^2 + 2L_x'\beta_{vx}\gamma^2 \leq 5L_v'\rho^2 \ . \tag{A.29}$$

With $\rho \leq \frac{\overline{\beta}_{vx}}{2\beta_{vx}}$ we get

$$2\beta_{vx}\gamma^2 + \overline{\beta}_{yx}\frac{\gamma^2}{\rho} \leq 2\overline{\beta}_{vx}\frac{\gamma^2}{\rho} \ . \tag{A.30}$$

As a consequence of Equations (A.26), (A.27), (A.28), (A.29) and (A.30), we have

$$\delta_v^{t+1} \leq \left(1 - \frac{\rho\mu_g}{16}\right)\delta_v^t + 3\beta_{vy}\rho\delta_y^t + 5L_v'\rho^2 S^t + 2\overline{\beta}_{vx}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \ .$$

For the inequality on Φ^t, we start from Equations (4.9) and (A.20)

$$\begin{aligned} h^{t+1} \leq \Phi^t - \frac{\gamma}{2}g^t - \left(\frac{\gamma}{2} - L_\Phi\gamma^2\right)\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \\ + \left(\frac{L_x^2}{2}\gamma + L_\Phi L_x''\gamma^2\right)(\delta_y^t + \delta_v^t) + L_\Phi L_x'\gamma^2 S^t \ . \end{aligned} \tag{A.31}$$

Assuming $\gamma \leq \min\left(\frac{1}{4L_\Phi}, \frac{L_x^2}{2L_\Phi L_x''}\right)$ leads

$$\Phi^{t+1} \leq \Phi^t - \frac{\gamma}{2}g^t - \frac{\gamma}{4}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] + L_x^2\gamma(\delta_y^t + \delta_v^t) + L_\Phi L_x'\gamma^2 S^t \ .$$

□

We are now ready to prove Theorem 4.3.

Proof. We consider the Lyapunov function

$$\mathcal{L}^t = \Phi^t + \phi_s S^t + \phi_y\delta_y^t + \phi_v\delta_v^t \tag{A.32}$$

for some constants ϕ_s, ϕ_y and ϕ_v.

We have

$$\begin{aligned} \mathcal{L}^{t+1} - \mathcal{L}^t \leq -\frac{\gamma}{2}g^t - \left(\frac{\gamma}{4} - 2\phi_y\overline{\beta}_{yx}\frac{\gamma^2}{\rho} - 2\phi_v\overline{\beta}_{vx}\frac{\gamma^2}{\rho} - \phi_s P\gamma^2\right)\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \\ - \left(\phi_y\frac{\mu_g}{8}\rho - L_x^2\gamma - 8\phi_v\beta_{vy}\rho - \phi_s\beta_{sy}\rho^2\right)\delta_y^t \\ - \left(\phi_v\frac{\mu_g}{16}\rho - L_x^2\gamma - 2\phi_y L_x''\gamma^2 - \phi_s\beta_{sv}\rho^2\right)\delta_v^t \\ - \left(\phi_s\frac{\Gamma}{2} - 5\phi_y L_y'\rho^2 - 5\phi_v L_v'\rho^2 - L_\Phi L_x'\gamma^2\right)S^t \ . \end{aligned}$$

To get a decrease, ϕ_y, ϕ_v and ϕ_s, ρ and γ must be such that:

$$\begin{aligned} 2\phi_y\overline{\beta}_{yx}\frac{\gamma^2}{\rho} + 2\phi_v\overline{\beta}_{vx}\frac{\gamma^2}{\rho} + \phi_s P\gamma^2 \leq \frac{\gamma}{4} \\ L_x^2\gamma + 8\phi_v\beta_{vy}\rho + \phi_s\beta_{sy}\rho^2 \leq \phi_y\frac{\mu_g}{8}\rho \\ L_x^2\gamma + 8\phi_y L_x''\gamma^2 + \phi_s\beta_{sv}\rho^2 \leq \phi_v\frac{\mu_g}{16}\rho \\ 5\phi_y L_y'\rho^2 + 5\phi_v L_v'\rho^2 + L_\Phi L_x'\gamma^2 \leq \phi_s\frac{\Gamma}{2} \ . \end{aligned}$$

In order to take into account the scaling of the quantities with respect to $N = n+m$, we take $\rho = \rho' N^{n_\rho}$, $\gamma = \gamma' N^{n_\gamma}$, $\phi_y = \phi'_y N^{n_y}$, $\phi_v = \phi'_v N^{n_v}$ and $\phi_s = \phi'_s N^{n_s}$. Since $\Gamma = \mathcal{O}(N^{-1})$, $P = \mathcal{O}(N)$, $\beta_{sy} = \mathcal{O}(N)$ and $\beta_{sv} = \mathcal{O}(N)$, we also define $\Gamma' = \Gamma N$, $P' = PN^{-1}$, $\beta'_{sy} = \beta_{sy}N^{-1}$ and $\beta'_{sv}N^{-1}$. Now, the previous Equations read (after slight simplifications):

$$
\begin{aligned}
(2\phi'_y\overline{\beta}_{yx} + 2\phi'_v\overline{\beta}_{vx})\frac{\gamma'}{\rho'}N^{n_y+n_\gamma-n_\rho} + \phi'_s P'\gamma' N^{n_s+n_\gamma+1} &\leq \frac{1}{4} \\
L_x^2\gamma' N^{n_\gamma} + 8\phi'_v\beta_{vy}\rho' N^{n_v+n_\rho} + \phi'_s\beta'_{sy}(\rho')^2 N^{2n_\rho+n_s+1} &\leq \phi'_y\frac{\mu_g}{8}\rho' N^{n_y+n_\rho} \\
L_x^2\gamma' N^{n_\gamma} + 8\phi'_y L''_x(\gamma')^2 N^{2n_\gamma+n_y} + \phi'_s\beta'_{sv}(\rho')^2 N^{n_s+2n_\rho+1} &\leq \phi'_v\frac{\mu_g}{16}\rho' N^{n_v+n_\rho} \\
5\phi'_y L'_y(\rho')^2 N^{n_y+2n_\rho} + 5\phi'_v L'_v(\rho')^2 N^{2n_\rho+n_v} + L_\Phi L'_x(\gamma')^2 N^{n_\gamma} &\leq \phi_s\frac{\Gamma'}{2}N^{n_s-1} \ .
\end{aligned}
$$

In order to ensure that the exponents on N are lower in the left-hand-side than those on the right-hand-side, we take $n_y = n_v = 0$, $n_\rho = n_\gamma = -\frac{2}{3}$ and $n_s = -\frac{1}{3}$. The Equations become

$$
\begin{aligned}
(2\phi'_y\overline{\beta}_{yx} + 2\phi'_v\overline{\beta}_{vx})\frac{\gamma'}{\rho'} + \phi'_s P'\gamma' &\leq \frac{1}{4} \\
L_x^2\gamma' N^{-2/3} + 8\phi'_v\beta_{vy}\rho' N^{-2/3} + \phi'_s\beta'_{sy}(\rho')^2 N^{-2/3} &\leq \phi'_y\frac{\mu_g}{8}\rho' N^{-2/3} \\
L_x^2\gamma' N^{-2/3} + 8\phi'_y L''_x(\gamma')^2 N^{-4/3} + \phi'_s\beta'_{sv}(\rho')^2 N^{-2/3} &\leq \phi'_v\frac{\mu_g}{16}\rho' N^{-2/3} \\
5\phi'_y L'_y(\rho')^2 N^{-4/3} + 5\phi'_v L'_v(\rho')^2 N^{-4/3} + L_\Phi L'_x(\gamma')^2 N^{-4/3} &\leq \phi'_s\frac{\Gamma'}{2}N^{-4/3} \ .
\end{aligned}
$$

We can replace the penultimate equation by the stronger

$$
L_x^2\gamma' N^{-2/3} + 8\phi'_y L''_x(\gamma')^2 N^{-2/3} + \phi'_s\beta'_{sv}(\rho')^2 N^{-2/3} \leq \phi'_v\frac{\mu_g}{16}\rho' N^{-2/3}
$$

so that we can simplify all the equations by dropping the dependencies in N:

$$
\begin{aligned}
(2\phi'_y\overline{\beta}_{yx} + 2\phi'_v\overline{\beta}_{vx})\frac{\gamma'}{\rho'} + \phi'_s P'\gamma' &\leq \frac{1}{4} \\
L_x^2\gamma' + 8\phi'_v\beta_{vy}\rho' + \phi'_s\beta'_{sy}(\rho')^2 &\leq \phi'_y\frac{\mu_g}{8}\rho' \\
L_x^2\gamma' + 8\phi'_y L''_x(\gamma')^2 + \phi'_s\beta'_{sv}(\rho')^2 &\leq \phi'_v\frac{\mu_g}{16}\rho' \\
5\phi'_y L'_y(\rho')^2 + 5\phi'_v L'_v(\rho')^2 + L_\Phi L'_x(\gamma')^2 &\leq \phi'_s\frac{\Gamma'}{2} \ .
\end{aligned}
$$

Let us take $\phi'_s = 1$, $\phi'_y = \phi''_y\frac{\rho'}{\gamma'}$ and $\phi'_v = \phi''_v\frac{\rho'}{\gamma'}$ with $\phi''_y = \frac{1}{32\overline{\beta}_{yx}}$ and $\phi''_v = \min\left(\frac{1}{32\overline{\beta}_{vx}}, \phi''_y\frac{\mu_g}{128\beta_{vy}}\right)$. The equations become

$$
\begin{aligned}
P'\gamma' &\leq \frac{1}{8} \\
L_x^2\gamma' + \beta'_{sy}(\rho')^2 &\leq \phi''_y\frac{\mu_g}{16}\frac{(\rho')^2}{\gamma'} \\
L_x^2\gamma' + 8\phi''_y L''_x\gamma'\rho' + \beta'_{sv}(\rho')^2 &\leq \phi''_v\frac{\mu_g}{16}\frac{(\rho')^2}{\gamma'} \\
5\phi''_y L'_y\frac{(\rho')^3}{\gamma'} + 5\phi''_v L'_v\frac{(\rho')^3}{\gamma'} + L_\Phi L'_x(\gamma')^2 &\leq \frac{\Gamma'}{2} \ .
\end{aligned}
$$

The condition $\gamma' \leq \frac{1}{8P'}$ ensures that the first equation is verified. With $\gamma' \leq \min\left(\sqrt{\frac{\phi''_y\mu_g}{32L_x^2}}\rho', \frac{\phi''_y\mu_g}{32\beta'_{sz}}\right)$, the second equations is verified. With $\gamma' \leq \min\left(\sqrt{\frac{\phi''_v\mu_g}{48L_x^2}}\rho', \frac{\phi''_v\mu_g}{48\beta'_{sv}}, \sqrt{\frac{\phi''_v\mu_g}{384\phi''_y L''_x\rho'}}\right)$, the third is verified. With $\gamma' \leq \sqrt{\frac{\Gamma'}{6L_\Phi L'_x}}$, the last can be simplified:

$$
(5\phi''_y L'_y + 5\phi''_v L'_v)(\rho')^3 \leq \frac{\Gamma'}{3}\gamma' \ .
$$

Let us write $\gamma' = \xi\rho'$. If we want that equation does no contradict the previous upper bound on γ' involving ρ' and the conditions of Lemma A.8, that is

$$\gamma' \leq \underbrace{\min\left(\sqrt{\frac{\phi''_y\mu_g}{32L_x^2}}, \sqrt{\frac{\phi''_v\mu_g}{48L_x^2}}, \sqrt{\frac{L'_y}{2L'_x\beta_{yx}}}, \sqrt{\frac{L'_v}{2L'_x\beta_{vx}}}\right)}_{K_1}\rho'$$

$$\gamma' \leq \underbrace{\min\left(\sqrt{\frac{\mu_g}{64\beta_{yx}L''_x}}, \sqrt{\frac{\mu_g}{128\beta_{vx}L''_x}}, \sqrt{\frac{\beta_{vy}}{4L''_x\beta_{vx}}}\right)}_{K_2}\sqrt{\rho'}$$

$$\gamma' \leq \underbrace{\sqrt{\frac{\phi''_v\mu_g}{384\phi''_yL''_x}}}_{K_3}\frac{1}{\sqrt{\rho'}}$$

$$\gamma' \leq \underbrace{\min\left(\frac{1}{4L_\Phi}, \frac{L_x^2}{2L_\Phi L''_x}, \sqrt{\frac{\Gamma'}{6L_\Phi L'_x}}, \frac{1}{8P'}, \frac{\phi''_y\mu_g}{32\beta'_{sz}}, \frac{\phi''_v\mu_g}{48\beta'_{sv}}\right)}_{K_4}$$

$$\gamma' \geq \underbrace{\frac{15(\phi''_yL'_y + \phi''_vL'_v)}{\Gamma'}}_{K_5}\rho^3$$

ξ must verify

$$\begin{aligned}
\xi &\leq K_1\\
\xi &\leq K_2(\rho')^{-\frac{1}{2}}\\
\xi &\leq K_3(\rho')^{-\frac{3}{2}}\\
\xi &\leq K_4(\rho')^{-1}\\
\xi &\geq K_5(\rho')^2
\end{aligned}$$

which is possible if ρ' satisfies

$$\rho' \leq \min\left(\sqrt{\frac{K_1}{K_5}}, \left(\frac{K_2}{K_5}\right)^{-\frac{3}{2}}, \left(\frac{K_3}{K_5}\right)^{-\frac{5}{2}}, \left(\frac{K_4}{K_5}\right)^{-2}\right) \ .$$

Let us take

$$\rho' = \min\left(\sqrt{\frac{K_1}{K_5}}, \left(\frac{K_2}{K_5}\right)^{-\frac{3}{2}}, \left(\frac{K_3}{K_5}\right)^{-\frac{5}{2}}, \left(\frac{K_4}{K_5}\right)^{-2}, \frac{\mu_g}{64L_y^2}, \frac{\overline{\beta}_{yx}}{2\beta_{yx}}, \frac{\mu_g}{128(L_v^2+L''_v)}, \frac{\beta_{vy}}{8(L_v^2+L''_v)}, \frac{\overline{\beta}_{vx}}{2\beta_{vx}}\right)$$

and

$$\xi = \min(K_1, K_2(\rho')^{-\frac{1}{2}}, K_3(\rho')^{-\frac{3}{2}}, K_4(\rho')^{-1}) \ .$$

Finally, we have

$$\mathcal{L}^{t+1} - \mathcal{L}^t \leq -\frac{\gamma}{2}g^t$$

and therefore, summing and telescoping yields

$$\frac{1}{T}\sum_{t=1}^{T} g^t \leq \frac{\mathcal{L}^1}{\gamma T} = \frac{\mathcal{L}^0 N^{\frac{2}{3}}}{T} \ .$$

Since with respect to N we have

$$\mathcal{L}^0 = \Phi^0 + \phi_y\delta_y^0 + \phi_v\delta_v^0 + \phi_s S^0 = \mathcal{O}(N^{-1} + 1 + 1 + N^{-\frac{1}{3}}) = \mathcal{O}(1) \ ,$$

we end up with

$$\boxed{\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla\Phi(x^t)\|^2] = \mathcal{O}\left(\frac{N^{\frac{2}{3}}}{T}\right) \ .}$$

□

A.1.6 Proof of Theorem 4.4

We are now going to prove Theorem 4.4 that we recall here:

Theorem 4.4 (Convergence of SABA, PL case)**.** *Assume that Φ satisfies the PL inequality and that Assumptions 4.1 to 4.3 and 4.5 to 4.6 hold. We suppose $\rho = \rho' N^{-\frac{2}{3}}$ and $\gamma = \xi\rho' N^{-1}$, where ρ' and ξ depend only on f and g and are specified in appendix. Let x^t the iterates of SABA and $c' \triangleq \min\left(\mu_\Phi, \frac{1}{16P'}\right)$ with P' specified in the appendix. Then,*

$$\mathbb{E}[\Phi^T] - \Phi^* = (1 - c'\gamma)^T(\Phi^0 - \Phi^* + C^0)$$

where C^0 is a constant specified in appendix that depends on the initialization of y, v, x and memory.

Here, we have

$$\rho' = \min\left(\sqrt{\frac{K_1'}{K_5}}, \left(\frac{K_2}{K_5}\right)^{\frac{2}{5}}, \left(\frac{K_3}{K_5}\right)^{\frac{2}{7}}, \left(\frac{K_4'}{K_5}\right)^{\frac{1}{3}}, \frac{\mu_g}{64L_y^2}, \frac{\overline{\beta}_{yx}}{2\beta_{yx}}, \frac{\mu_g}{128(L_v^2 + L_v'')}, \frac{\beta_{vy}}{8(L_v^2 + L_v'')}, \frac{\overline{\beta}_{vx}}{2\beta_{vx}}\right) \ ,$$

and

$$\xi = \min(K_1', K_2(\rho')^{-\frac{1}{2}}, K_3(\rho')^{-\frac{3}{2}}, K_4'(\rho')^{-1}) \ .$$

where $P' = PN^{-1}$, $\Gamma' = \Gamma N$,

$$\phi_y'' = \frac{1}{32\overline{\beta}_{yx}} \ , \phi_v'' = \min\left(\frac{1}{32\overline{\beta}_{vx}}, \phi_y'' \frac{\mu_g}{128\beta_{vy}}\right) \ ,$$

$$K_1' = \min\left(\frac{\mu_g}{64c'}, \sqrt{\frac{\phi_y''\mu_g}{48L_x^2}}, \sqrt{\frac{\phi_v''\mu_g}{64L_x^2}}, \sqrt{\frac{L_y'}{2L_x'\beta_{yx}}}, \sqrt{\frac{L_v'}{2L_x'\beta_{vx}}}\right) \ ,$$

$$K_2 = \min\left(\sqrt{\frac{\mu_g}{64\beta_{yx}L_x''}}, \sqrt{\frac{\mu_g}{128\beta_{vx}L_x''}}, \sqrt{\frac{\beta_{vy}}{4L_x''\beta_{vx}}}\right) \ ,$$

$$K_3 = \sqrt{\frac{\phi_v''\mu_g}{512\phi_y''L_x''}} \ , \quad K_4' = \min\left(\frac{\Gamma'}{6c'}, \frac{1}{4L_\Phi}, \frac{L_x^2}{2L_\Phi L_x''}, \sqrt{\frac{\Gamma'}{6L_\Phi L_x'}}, \frac{1}{18P'}, \frac{\phi_y''\mu_g}{48\beta_{sz}'}, \frac{\phi_v''\mu_g}{64\beta_{sv}'}\right)$$

and

$$K_5 = \frac{20(\phi_y''L_y' + \phi_v''L_v')}{\Gamma'} \ .$$

Proof. For simplicity, we assume that $\Phi^* = 0$ and so for any $x \in \mathbb{R}^{d_x}$ the PL inequality reads:

$$\frac{1}{2}\|\nabla\Phi(x)\|^2 \geq \mu_\Phi\Phi(x) \ .$$

Then, eq. (A.21) gives

$$h^{t+1} \leq \left(1 - \frac{\gamma\mu_h}{2}\right)\Phi^t - \frac{\gamma}{4}\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] + \gamma L_x^2(\delta_y^t + \delta_v^t) + L_\Phi L_x'\gamma^2 S^t \ .$$

We take $\mathcal{L}^t$ the Lyapunov function given in Equation (A.32). We find

$$\begin{aligned}\mathcal{L}^{t+1} - \mathcal{L}^t \leq &-\gamma\mu_h\Phi^t - \left(\frac{\gamma}{4} - 2\phi_y\overline{\beta}_{yx}\frac{\gamma^2}{\rho} - 2\phi_v\overline{\beta}_{vx}\frac{\gamma^2}{\rho} - \phi_s P\gamma^2\right)\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \\ &- \left(\phi_y\frac{\mu_g}{8}\rho - L_x^2\gamma - 8\phi_v\beta_{vy}\rho - \phi_s\beta_{sy}\rho^2\right)\delta_y^t \\ &- \left(\phi_v\frac{\mu_g}{16}\rho - L_x^2\gamma - 2\phi_y L_x''\gamma^2 - \phi_s\beta_{sv}\rho^2\right)\delta_v^t \\ &- \left(\phi_s\frac{\Gamma}{2} - 5\phi_y L_y'\rho^2 - 5\phi_v L_v'\rho^2 - L_\Phi L_x'\gamma^2\right)S^t \ .\end{aligned}$$

We now try to find linear convergence, hence we add to this $c\mathcal{L}^t$ to get

$$\begin{aligned}\mathcal{L}^{t+1} - (1-c)\mathcal{L}^t \leq & -(\gamma\mu_h - c)\Phi^t - \left(\frac{\gamma}{4} - 2\phi_y\overline{\beta}_{yx}\frac{\gamma^2}{\rho} - 2\phi_v\overline{\beta}_{vx}\frac{\gamma^2}{\rho} - \phi_s P\gamma^2 - c\right)\mathbb{E}[\|D_x(y^t, v^t, x^t)\|^2] \\ & - \left(\phi_y\frac{\mu_g}{8}\rho - L_x^2\gamma - 8\phi_v\beta_{vy}\rho - \phi_s\beta_{sy}\rho^2 - c\phi_y\right)\delta_y^t \\ & - \left(\phi_v\frac{\mu_g}{16}\rho - L_x^2\gamma - 2\phi_y L_x''\gamma^2 - \phi_s\beta_{sv}\rho^2 - c\phi_v\right)\delta_v^t \\ & - \left(\phi_s\frac{\Gamma}{2} - 5\phi_y L_y'\rho^2 - 5\phi_v L_v'\rho^2 - L_\Phi L_x'\gamma^2 - c\phi_S\right)S^t \ .\end{aligned}$$

Hence, the set of inequations for decrease becomes

$$\begin{aligned} c &\leq \gamma\mu_h \\ 2\phi_y\overline{\beta}_{yx}\frac{\gamma^2}{\rho} + 2\phi_v\overline{\beta}_{vx}\frac{\gamma^2}{\rho} + \phi_s P\gamma^2 + c &\leq \frac{\gamma}{4} \\ L_x^2\gamma + 8\phi_v\beta_{vy}\rho + \phi_s\beta_{sy}\rho^2 + \phi_y c &\leq \phi_y\frac{\mu_g}{8}\rho \\ L_x^2\gamma + 8\phi_y L_x''\gamma^2 + \phi_s\beta_{sv}\rho^2 + \phi_v c &\leq \phi_v\frac{\mu_g}{16}\rho \\ 5\phi_y L_y'\rho^2 + 5\phi_v L_v'\rho^2 + L_\Phi L_x'\gamma^2 + \phi_s c &\leq \phi_s\frac{\Gamma}{2} \ . \end{aligned}$$

We see that it is more convenient to write $c = \gamma c'$. As previously, we write $\gamma = \gamma' N^{n_\gamma}$, $\rho = \rho' N^{n_\rho}$, $\phi_y = \phi_y' N^{n_y}$, $\phi_v = \phi_v' N^{n_v}$, $\phi_s = \phi_s' N^{n_s}$, $P = P'N$, $\Gamma = \Gamma' N^{-1}$, $\beta_{sx} = \beta_{sx}' N$ and $\beta_{sv} = \beta_{sv}' N$. The equations read:

$$\begin{aligned} c' &\leq \mu_h \\ 2\phi_y'\overline{\beta}_{yx}\frac{\gamma'}{\rho'}N^{n_y+n_\gamma-n_\rho} + 2\phi_v'\overline{\beta}_{vx}\frac{\gamma'}{\rho'}N^{n_v+n_\gamma-n_\rho} + \phi_s' P'\gamma' N^{n_s+1+n_\gamma} + c' &\leq \frac{1}{4} \\ L_x^2\gamma' N^{n_\gamma} + 8\phi_v'\beta_{vy}\rho' N^{n_v+n_\rho} + \phi_s'\beta_{sy}'(\rho')^2 N^{n_s+2n_\rho+1} + \phi_y' c'\gamma' N^{n_y+n_\gamma} &\leq \phi_y'\frac{\mu_g}{8}\rho' N^{n_\rho+n_y} \\ L_x^2\gamma' N^{n_\gamma} + 8\phi_y' L_x''(\gamma')^2 N^{n_y+2n_\gamma} + \phi_s'\beta_{sv}'(\rho')^2 N^{n_s+1+2n_\rho} + \phi_v' c'\gamma' N^{n_v+n_\gamma} &\leq \phi_v'\frac{\mu_g}{16}\rho' N^{n_v+n_\rho} \\ 5\phi_y' L_y'(\rho')^2 N^{n_y+2n_\rho} + 5\phi_v' L_v'(\rho')^2 N^{n_v+2n_\rho} + L_\Phi L_x'(\gamma')^2 N^{2n_\gamma} + \phi_s' c'\gamma' N^{n_s+n_\gamma} &\leq \phi_s'\frac{\Gamma'}{2}N^{n_s-1} \ . \end{aligned}$$

In order to ensure that the exponents on N are lower in the left-hand-side than those on the right-hand-side, we take $n_y = n_v = 0$, $n_\rho = -\frac{2}{3}$, $n_\gamma = -1$ and $n_s = -\frac{1}{3}$. The Equations become

$$\begin{aligned} c' &\leq \mu_h \\ 2\phi_y'\overline{\beta}_{yx}\frac{\gamma'}{\rho'}N^{-\frac{1}{3}} + 2\phi_v'\overline{\beta}_{vx}\frac{\gamma'}{\rho'}N^{-\frac{1}{3}} + \phi_s' P'\gamma' N^{-\frac{1}{3}} + c' &\leq \frac{1}{4} \\ L_x^2\gamma' N^{-1} + 8\phi_v'\beta_{vy}\rho' N^{-\frac{2}{3}} + \phi_s'\beta_{sy}'(\rho')^2 N^{-\frac{2}{3}} + \phi_y' c'\gamma' N^{-1} &\leq \phi_y'\frac{\mu_g}{8}\rho' N^{-\frac{2}{3}} \\ L_x^2\gamma' N^{-1} + 8\phi_y' L_x''(\gamma')^2 N^{-2} + \phi_s'\beta_{sv}'(\rho')^2 N^{-\frac{2}{3}} + \phi_v' c'\gamma' N^{-1} &\leq \phi_v'\frac{\mu_g}{16}\rho' N^{-\frac{2}{3}} \\ 5\phi_y' L_y'(\rho')^2 N^{-\frac{4}{3}} + 5\phi_v' L_v'(\rho')^2 N^{-2} + L_\Phi L_x'(\gamma')^2 N^{-2} + \phi_s' c'\gamma' N^{-\frac{4}{3}} &\leq \phi_s'\frac{\Gamma'}{2}N^{-\frac{4}{3}} \ . \end{aligned}$$

Now we have to find ρ', γ', ϕ_y', ϕ_v' and ϕ_s' that verifies the following conditions (which are a bit stronger than thoose in the previous Equations):

$$\begin{aligned} c' &\leq \mu_h \\ 2\phi_y'\overline{\beta}_{yx}\frac{\gamma'}{\rho'} + 2\phi_v'\overline{\beta}_{vx}\frac{\gamma'}{\rho'} + \phi_s' P'\gamma' + c' &\leq \frac{1}{4} \\ L_x^2\gamma' + 8\phi_v'\beta_{vy}\rho' + \phi_s'\beta_{sy}'(\rho')^2 + \phi_y' c'\gamma' &\leq \phi_y'\frac{\mu_g}{8}\rho' \\ L_x^2\gamma' + 8\phi_y' L_x''(\gamma')^2 + \phi_s'\beta_{sv}'(\rho')^2 + \phi_v' c'\gamma' &\leq \phi_v'\frac{\mu_g}{16}\rho' \\ 5\phi_y' L_y'(\rho')^2 + 5\phi_v' L_v'(\rho')^2 + L_\Phi L_x'(\gamma')^2 + \phi_s' c'\gamma' &\leq \phi_s'\frac{\Gamma'}{2} \ . \end{aligned}$$

As previously, we take $\phi'_s = 1$ and we denote $\phi'_z = \phi''_y \frac{\rho'}{\gamma'}$ with $\phi''_y = \frac{1}{32\overline{\beta}_{yx}}$ and $\phi'_z = \phi''_y \frac{\rho'}{\gamma'}$ with $\phi''_v = \min\left(\frac{1}{32\overline{\beta}_{vx}}, \phi''_y \frac{\mu_g}{128\beta_{vy}}\right)$, the equations become

$$\begin{aligned}
c' &\leq \mu_h \\
P'\gamma' + c' &\leq \frac{1}{8} \\
L_x^2(\gamma')^2 + \beta'_{sy}(\rho')^2\gamma' + \phi''_y c'\rho'\gamma' &\leq \phi''_y \frac{\mu_g}{16}(\rho')^2 \\
L_x^2(\gamma')^2 + 8\phi''_y L''_x \rho'(\gamma')^2 + \beta'_{sv}(\rho')^2\gamma' + \phi''_v c'\rho'\gamma' &\leq \phi''_v \frac{\mu_g}{16}(\rho')^2 \\
5\phi''_y L'_y(\rho')^3 + 5\phi''_v L'_v(\rho')^3 + L_\Phi L'_x(\gamma')^3 + c'(\gamma')^2 &\leq \frac{\Gamma'}{2}\gamma' \ .
\end{aligned}$$

Since $c' \leq \frac{1}{16}$ and $\gamma' \leq \frac{1}{16P'}$, the second equation is verified. With $\gamma' \leq \min\left(\sqrt{\frac{\phi''_y \mu_g}{48L_x^2}}\rho', \frac{\phi''_y \mu_g}{48\beta_{sv}}\right)$ and $c' \leq \frac{\mu_g \rho'}{48\gamma'}$ the third is verified. The conditions $\gamma' \leq \min\left(\sqrt{\frac{\phi''_v \mu_g}{64L_x^2}}\rho', \sqrt{\frac{\phi''_v \mu_g}{512\phi''_y L''_x \rho'}}, \frac{\phi''_v \mu_g}{64\beta'_{sv}}\right)$ and $c' \leq \frac{\mu_g \rho'}{64\gamma'}$ ensure that the forth is verified. With $\gamma' \leq \sqrt{\frac{\Gamma'}{8L_\Phi L'_x}}$ and $c' \leq \frac{\Gamma'}{8\gamma'}$, the fifth is simplified in

$$5\phi''_y L'_y(\rho')^3 + 5\phi''_v L'_v(\rho')^3 \leq \frac{\Gamma'}{4}\gamma' \ .$$

As in the proof of Theorem 4.3, let us denote $\gamma' = \xi\rho'$. To verify this equation and the previous bounds on γ' and c', we need

$$\begin{aligned}
\gamma' &\leq \underbrace{\min\left(\sqrt{\frac{\phi''_y \mu_g}{48L_x^2}}, \sqrt{\frac{\phi''_v \mu_g}{64L_x^2}}, \sqrt{\frac{L'_y}{2L'_x\beta_{yx}}}, \sqrt{\frac{L'_v}{2L'_x\beta_{yx}}}\right)}_{K_1}\rho' \ , \\
\gamma' &\leq \underbrace{\min\left(\sqrt{\frac{\mu_g}{64\beta_{yx}L''_x}}, \sqrt{\frac{\mu_g}{128\beta_{vx}L''_x}}, \sqrt{\frac{\beta_{vy}}{4L''_x\beta_{vx}}}\right)}_{K_2}\sqrt{\rho'} \ , \\
\gamma' &\leq \underbrace{\sqrt{\frac{\phi''_v \mu_g}{512\phi''_y L''_x}}}_{K_3}\frac{1}{\sqrt{\rho'}} \ , \\
\gamma' &\leq \underbrace{\min\left(\frac{1}{4L_\Phi}, \frac{L_x^2}{2L_\Phi L''_x}, \frac{\phi''_y \mu_g}{48\beta_{sv}}, \frac{\phi''_v \mu_g}{64\beta'_{sv}}, \frac{1}{16P'}, \sqrt{\frac{\Gamma'}{8L_\Phi L'_x}}\right)}_{K_4} \\
\gamma' &\geq \underbrace{\frac{20(\phi''_y L'_y + \phi''_v L'_v)}{20}}_{K_5}(\rho')^3 \ , \\
c' &\leq \underbrace{\min\left(\mu_h, \frac{1}{16}, \frac{1}{16P'}\right)}_{K_6} \ , \\
c' &\leq \underbrace{\frac{\mu_g}{64}}_{K_7}\frac{1}{\xi} \ , \\
c' &\leq \underbrace{\frac{\Gamma'}{8}}_{K_8}\frac{1}{\gamma'} \ .
\end{aligned}$$

So, ξ, ρ' and c' must verify

$$\begin{aligned}
\xi &\leq \underbrace{\min\left(K_1, \frac{K_7}{c'}\right)}_{K_1'} \ , \\
\xi &\leq K_2(\rho')^{-\frac{1}{2}} \ , \\
\xi &\leq K_3(\rho')^{-\frac{3}{2}} \ , \\
\xi &\leq \underbrace{\min\left(K_4, \frac{K_8}{c'}\right)}_{K_4'}(\rho')^{-1} \\
\xi &\geq K_5(\rho')^2 \ , \\
c' &\leq \underbrace{\min\left(\mu_h, \frac{1}{16}, \frac{1}{16P'}\right)}_{K_6} \ ,
\end{aligned}$$

which is possible if

$$\rho' \leq \min\left(\sqrt{\frac{K_1'}{K_5}}, \left(\frac{K_2}{K_5}\right)^{\frac{2}{5}}, \left(\frac{K_3}{K_5}\right)^{\frac{2}{7}}, \left(\frac{K_4'}{K_5}\right)^{\frac{1}{3}}\right) \ .$$

So let us take $c' = \min\left(\mu_h, \frac{1}{16}, \frac{1}{16P'}\right) = \min\left(\mu_h, \frac{1}{16P'}\right)$,

$$\rho' = \min\left(\sqrt{\frac{K_1'}{K_5}}, \left(\frac{K_2}{K_5}\right)^{\frac{2}{5}}, \left(\frac{K_3}{K_5}\right)^{\frac{2}{7}}, \left(\frac{K_4'}{K_5}\right)^{\frac{1}{3}}, \frac{\mu_g}{64L_y^2}, \frac{\overline{\beta}_{yx}}{2\beta_{yx}}, \frac{\mu_g}{128(L_v^2+L_v'')}, \frac{\beta_{vy}}{8(L_v^2+L_v'')}, \frac{\overline{\beta}_{vx}}{2\beta_{vx}}\right)$$

and

$$\xi = \min(K_1, K_2(\rho')^{-\frac{1}{2}}, K_3(\rho')^{-\frac{3}{2}}, K_4(\rho')^{-1}) \ .$$

We have

$$\mathcal{L}^{t+1} \leq (1-c)\mathcal{L}^t$$

therefore, unrolling yields

$$\boxed{\Phi^t - \Phi^* \leq \mathcal{L}^t \leq (1-c'\gamma)^t\mathcal{L}^0.}$$

□

A.2 Convergence rates with weaker regularity assumptions

To get our rates, we need stronger assumptions than in the stochastic bilevel optimization literature Ghadimi and Wang (2018); Hong et al. (2023); Ji et al. (2021); Arbel and Mairal (2022a). In this section, we shortly present the convergence rates we can expect if we replace Assumptions 4.1 and 4.2 by Assumptions A.1 and A.2.

Assumption A.1. *The function f is differentiable. The gradient ∇f is Lipschitz continuous in (x, y) with Lipschitz constants $L_{f,1}$.*

Assumption A.2. *The function g is twice continuously differentiable on $\mathbb{R}^{d_y} \times \mathbb{R}^{d_x}$. For any $x \in \mathbb{R}^{d_x}$, $g(\cdot, x)$ is μ_g-strongly convex. The derivatives ∇g are $\nabla^2 g$ are Lipschitz continuous in (x, y) with respective Lipschitz constants $L_{g,1}$ and $L_{g,2}$.*

With these assumptions, we are not ensured that v^* is smooth, and so the descent lemmas take the form of Lemma A.9.

Lemma A.9. *Assume that $\rho \leq \frac{2}{\mu_g}$. We have:*

$$\delta_y^{t+1} \leq \left(1 - \frac{\rho\mu_g}{2}\right)\delta_y^t + 2\rho^2 V_y^t + 4\frac{L_*^2}{\mu_g}\frac{\gamma^2}{\rho}V_x^t$$
$$\delta_v^{t+1} \leq \left(1 - \frac{\rho\mu_g}{4}\right)\delta_v^t + \rho\beta_{vy}\delta_y^t + 2\rho^2 V_v^t + 8\frac{L_*^2}{\mu_g}\frac{\gamma^2}{\rho}V_x^t$$

where L_* is the maximum between the Lipschitz constants of y^* and v^* (see Lemma A.1) and $\beta_{vy} = \frac{1}{\mu_g^3}(L^F\mu_g + L_2^G)^2$.

Proof. **Inequality for δ_y.**

Instead of expanding the square as done in the proof of Lemma 4.2 in Equation (A.2), we use Young's inequality for some $a > 0$

$$\delta_y^{t+1} \leq (1+a)\mathbb{E}[\|y^{t+1} - y^*(x^t)\|^2] + (1+a^{-1})\mathbb{E}[\|y^*(x^{t+1}) - y^*(x^t)\|^2] \ .$$

Treating $\mathbb{E}[\|y^{t+1} - y^*(x^t)\|^2]$ and $\mathbb{E}[\|y^*(x^{t+1}) - y^*(x^t)\|^2]$ as done in the proof of Lemma 4.2 leads to

$$\delta_y^{t+1} \leq (1+a)\left[(1-\rho\mu_g)\delta_y^t + \rho^2 V_y^t\right] + (1+a^{-1})L_*^2\gamma^2 V_x^t$$

In order to keep a decrease in δ_y, we might want to use $a = \frac{1}{2}\rho\mu_g$, which gives the bound

$$\boxed{\delta_y^{t+1} \leq \left(1 - \frac{\rho\mu_g}{2}\right)\delta_y^t + 2\rho^2 V_y^t + \beta_{yx}\frac{\gamma^2}{\rho}V_x^t}$$

with $\beta_{yx} = 4\frac{L_*^2}{\mu_g}$. Indeed, this gives $(1 + \frac{1}{2}\rho\mu_g)(1 - \rho\mu_g) \leq 1 - \frac{1}{2}\rho\mu_g$. We have $a \leq 1$ since $\rho \leq \frac{2}{\mu_g}$, so $(1+a)\rho^2 \leq 2\rho^2$. Finally, we also have $1 + a^{-1} \leq 2a^{-1} = \frac{4}{\rho\mu_g}$.

Inequality for δ_v. As for δ_y, the difference with the proof of Lemma 4.2 is that we use we use Young's inequality for some $b > 0$ to get

$$\delta_v^{t+1} \leq (1+b)\mathbb{E}[\|v^{t+1} - v^*(x^t)\|^2] + (1+b^{-1})\mathbb{E}[\|v^*(x^{t+1}) - v^*(x^t)\|^2] \ .$$

The remaining part of the proof is similar to the proof of Lemma 4.2. □

The main difference with Lemma 4.2 is that we have $\mathcal{O}(\frac{\gamma^2}{\rho})$ in factor of V_x^t instead of $\mathcal{O}(\gamma^2)$. As a consequence, we need that the ratio $\frac{\gamma}{\rho}$ goes to zero to get convergence, as in Hong et al. (2023). This prevent us in getting rates that match rates of single level algorithms.

Hence, for SOBA, we have to choose $\gamma = \mathcal{O}(T^{-\frac{3}{5}})$ and $\rho = \mathcal{O}(T^{-\frac{2}{5}})$ and we end up with a convergence rate in $O(T^{-\frac{2}{5}})$. For SABA, we get a $\mathcal{O}((n+m)\epsilon^{-1})$ sample complexity, which is actually the sample complexity of SOBA used with full batch estimated directions.

CHAPTER 5

COMPLEXITY BOUNDS FOR BILEVEL EMPIRICAL RISK MINIMIZATION

Sommaire

This section presents the work published in AISTATS2024:

M. Dagréou, T. Moreau, S. Vaiter, and P. Ablin. A Lower Bound and a Near-Optimal Algorithm for Bilevel Empirical Risk Minimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024

Only minor modifications have been made from the original paper: adaptation of the notation, shorter introduction, relocation of some proofs in the main text, or addition of proof sketch for results with proof in the appendix (Chapter B).

5.1 Introduction

In this chapter, we build on the previous chapter's setting, aiming to solve:

$$\begin{aligned} &\min_{x\in\mathbb{R}^{d_x}} \Phi(x) = f(x, y^*(x)) \ , \\ &\text{subject to } y^*(x) \in \arg\min_{y\in\mathbb{R}^{d_y}} g(x,y) \end{aligned} \tag{5.1}$$

where the outer function f and the inner function g are empirical means:

$$f(x,y) = \frac{1}{m}\sum_{j=1}^{m} f_j(x,y), \quad g(x,y) = \frac{1}{n}\sum_{i=1}^{n} g_i(x,y) \ .$$

Previously, we demonstrated that in the non-convex/strongly convex setting, the SOBA and SABA algorithms achieve complexities in $\mathcal{O}\left(\epsilon^{-2}\right)$ and $\mathcal{O}\left((n+m)^{\frac{2}{3}}\epsilon^{-1}\right)$ respectively. These complexities are analogous to those of SGD (Robbins and Monro, 1951; Ghadimi and Lan, 2013) and SAGA (Defazio et al., 2014; Reddi et al., 2016) respectively for non-convex single-level optimization problems. This is despite the hypergradient bias arising from approximations of the inner problem solution $y^*(x)$ and the linear system solution $v^*(x)$. This suggests a possible transfer of complexity bounds from single-level to bilevel problems.

However, in classical single-level optimization, it is known that neither of these algorithms is optimal: the SARAH algorithm (Nguyen et al., 2017) achieves a better sample complexity of $\mathcal{O}(m^{\frac{1}{2}}\varepsilon^{-1})$ with m the number of samples. Furthermore, this algorithm is *near-optimal* (*i.e.* optimal up to constant factors) because the lower bound for single-level non-convex optimization is $\Omega(m^{\frac{1}{2}}\varepsilon^{-1})$ as proved by Zhou and Gu (2019). It is natural to ask if we can extend these results to bilevel optimization:

Are the optimal complexity bounds for solving bilevel optimization the same as in single-level optimization?

In single-level optimization, the problem of finding complexity lower bounds has been widely studied since the seminal work of Nemirovsky and Yudin (1983). On the one hand, Agarwal and Bottou (2015) provided a lower bound to minimize strongly convex and smooth finite sums with deterministic algorithms that have access to individual gradients. These results were extended to randomized algorithms for (strongly) convex finite sum objective by Woodworth and Srebro (2016). On the other hand, Carmon et al. (2020) provided a lower bound for minimizing non-convex functions with deterministic and randomized algorithms. The non-convex finite sum case is treated by Fang et al. (2018) and Zhou and Gu (2019). In the bilevel case, Ji and Liang (2023) showed a lower bound for deterministic algorithms. However, this result is restricted to the case where the value function Φ is convex or strongly convex, which is not the case with most ML-related bilevel problems. Our results are instead in a non-convex setting.

Contributions of Chapter 5. In Section 5.2, we introduce SRBA, an adaptation of the SARAH algorithm to the bilevel setting. We then demonstrate in Section 5.3 that, similarly to the single-level setting,

$$\mathcal{O}\left((n+m)^{\frac{1}{2}}\varepsilon^{-1} \vee (n+m)\right)$$

oracle calls are sufficient to reach an ε-stationary point. As shown in Table 5.1, it achieves the best-known complexity in the regime $n+m \lesssim \mathcal{O}(\varepsilon^{-2})$. In Section 5.4, we analyze the lower bounds for such problems. We show that we need at least $\Omega(m^{\frac{1}{2}}\varepsilon^{-1})$ oracle calls to reach an ε-stationary point (see Definition 5.1), hereby matching the previous upper-bound in the case where $n \asymp m$ and $\varepsilon \leq m^{-\frac{1}{2}}$. SRBA is thus near-optimal in that regime. Even though our main contribution is theoretical, we illustrate the numerical performances of the algorithm in Section 5.6.

5.2 A Near-Optimal Algorithm for Bilevel Empirical Risk Minimization

In this section, we introduce SRBA (Stochastic Recursive Bilevel Algorithm), a novel algorithm for bilevel empirical risk minimization which is provably near-optimal for this problem. This algorithm is inspired by the algorithms SPIDER (Fang et al., 2018) and SARAH (Nguyen et al., 2017, 2022) which are known for being near-optimal algorithms for non-convex finite sum minimization problems. It relies on a recursive estimation of directions of interest, which is restarted periodically.

	Complexity	Stochastic setting	f	g
StocBiO (Ji et al., 2021)	$\tilde{\mathcal{O}}(\varepsilon^{-2})$	General expectation	$\mathcal{C}_L^{1,1}$	SC and $\mathcal{C}_L^{2,2}$
AmIGO (Arbel and Mairal, 2022a)	$\mathcal{O}(\varepsilon^{-2})$	General expectation	$\mathcal{C}_L^{1,1}$	SC and $\mathcal{C}_L^{2,2}$
MRBO (Yang et al., 2021)	$\tilde{\mathcal{O}}(\varepsilon^{-\frac{3}{2}})$	General expectation	$\mathcal{C}_L^{1,1}$	SC and $\mathcal{C}_L^{2,2}$
VRBO (Yang et al., 2021)	$\tilde{\mathcal{O}}(\varepsilon^{-\frac{3}{2}})$	General expectation	$\mathcal{C}_L^{1,1}$	SC and $\mathcal{C}_L^{2,2}$
SABA (Dagréou et al., 2022a)	$\mathcal{O}((n+m)^{\frac{2}{3}}\varepsilon^{-1})$	Finite sum	$\mathcal{C}_L^{2,2}$	SC and $\mathcal{C}_L^{3,3}$
F^{2}SA (Kwon et al., 2023a)	$\mathcal{O}(\varepsilon^{-\frac{7}{2}})$	General expectation	$\mathcal{C}_L^{2,2}$	SC and $\mathcal{C}_L^{2,2}$
SRBA	$\mathcal{O}((n+m)^{\frac{1}{2}}\varepsilon^{-1})$	Finite sum	$\mathcal{C}_L^{2,2}$	SC and $\mathcal{C}_L^{3,3}$

Table 5.1: Comparison between the sample complexities and the Assumptions of some stochastic bilevel solvers. It corresponds to the number of calls to gradient, Hessian-vector products, and Jacobian-vector product sufficient to get an ε-stationary point. The tilde on the $\tilde{\mathcal{O}}$ hide a factor $\log(\varepsilon^{-1})$. "SC" means "strongly convex". $\mathcal{C}_L^{p,k}$ means p-times differentiable with Lipschitz kth order derivatives for $k \leq p$.

5.2.1 Assumptions

Before presenting our algorithm, we formulate regularity Assumptions on the functions f and g.

Assumption 5.1. *For all $j \in [m]$, f_j is twice differentiable and $L_{f,0}$-Lipschitz continuous. Its gradient is $L_{f,1}$-Lipschitz continuous and its Hessian is $L_{f,2}$-Lipschitz continuous.*

Assumption 5.2. *For all $i \in [n]$, g_i is three times differentiable. Its first, second, and third order derivatives are respectively $L_{g,1}$-Lipschitz continuous, $L_{g,2}$-Lipschitz continuous, and $L_{g,3}$-Lipschitz continuous. For $x \in \mathbb{R}^{d_x}$, the function $g_i(x, \cdot)$ is μ_g-strongly convex.*

The strong convexity and the smoothness with respect to y hold, for instance, when we consider an ℓ^2-regularized logistic regression problem with non-separable data. These regularity assumptions up to first-order for f and second-order for g are standard in the stochastic bilevel literature (Arbel and Mairal, 2022a; Ji et al., 2021; Yang et al., 2021). The second-order regularity for f and third-order regularity for g assumptions are necessary for the analysis of the dynamics of the auxiliary variable v we introduce in subsection 5.2.2, as is the case in the previous chapter. As shown by Ghadimi and Wang (2018, Lemma 2.2), these assumptions imply the smoothness of Φ as proved in Proposition 3.2, which is a fundamental property to get a descent. Another consequence of Assumptions 5.1 and 5.2 is the boundedness of the function v^*.

Proposition 5.1. *Assume that Assumptions 5.1 and 5.2 hold. Then, for $R = \frac{L_{f,0}}{\mu_g}$ it holds that for any $x \in \mathbb{R}^{d_x}$, we have $\|v^*(x)\| \leq R$.*

We denote Γ the closed ball centered in 0 with radius R, and Π_Γ the projection onto Γ. We also denote $\Pi(y, v, x) = (y, \Pi_\Gamma(v), x)$, for $(y, v, x) \in \mathbb{R}^{d_y} \times \mathbb{R}^{d_y} \times \mathbb{R}^{d_x}$.

5.2.2 Hypergradient Approximation

Recall that the gradient of Φ is given by

$$\nabla\Phi(x) = \nabla_x f(x, y^*(x)) + \nabla^2_{xy} g(x, y^*(x)) v^*(x)$$

where $v^*(x)$ is the solution of a linear system:

$$v^*(x) = -\left[\nabla^2_{yy} g(x, y^*(x))\right]^{-1} \nabla_y f(x, y^*(x)) \ . \tag{5.2}$$

This gradient is intractable in practice because it requires the perfect knowledge of $y^*(x)$ and $v^*(x)$ which are usually costly to compute. As classically done in the stochastic bilevel literature (Ji et al., 2021; Arbel and Mairal, 2022a; Li et al., 2022), $y^*(x)$ and $v^*(x)$ are replaced by approximate surrogate variables y and v. The variable y is typically the output of one or several steps of an optimization

procedure applied to $g(x, \cdot)$. The variable v can be computed by using Neumann approximations or doing some optimization steps on the quadratic $v \mapsto \frac{1}{2} v^\top \nabla^2_{yy} g(x,y) v + \nabla_y f(x,y)^\top v$. We consider the approximate hypergradient given by

$$D_x(y, v, x) = \nabla^2_{xy} g(x,y) v + \nabla_x f(x,y) \ .$$

The motivation behind this direction is that if we take $y = y^*(x)$ and $v = v^*(x)$, we recover the true gradient, that is $D_x(y^*(x), v^*(x), x) = \nabla \Phi(x)$. Proposition 5.2 controls the hypergradient approximation error by the distances between y and $y^*(x)$ and between v and $v^*(x)$ (see Lemma 4.1 for a proof).

Proposition 5.2. *Let $x \in \mathbb{R}^{d_x}$. Assume that f is differentiable and $L_{f,1}$ smooth with bounded gradient, g is twice differentiable with Lipschitz gradient and Hessian and $g(x, \cdot)$ is μ_g-strongly convex. Then there exists a constant L_x such that*

$$\|D_x(y, v, x) - \nabla \Phi(x)\|^2 \leq L_x^2 (\|y - y^*(x)\|^2 + \|v - v^*(x)\|^2).$$

Thus, it is natural to make y and v move towards their respective equilibrium values which are given by $y^*(x)$ and $v^*(x)$. As a consequence, we also introduce the directions D_y and D_x as follows

$$\begin{aligned} D_y(y, v, x) &= \nabla_y g(x,y) \ , \\ D_v(y, v, x) &= \nabla^2_{yy} g(x,y) v + \nabla_y f(x,y) \ . \end{aligned}$$

The interest of considering the directions D_y and D_v is recalled in Proposition 5.3 demonstrated in the previous chapter (see Proposition 4.1).

Proposition 5.3. *Assume that g is strongly convex with respect to its first variable. Then for any $x \in \mathbb{R}^{d_x}$, it holds $D_y(y^*(x), v^*(x), x) = 0$ and $D_v(y^*(x), v^*(x), x) = 0$.*

The directions D_y, D_v, and D_x can be written as sums over the samples. Hence, following these directions enables to adapt any classical algorithm suited for single-level finite sum minimization to bilevel finite sum minimization. In what follows, for two indices $i \in [n]$ and $j \in [m]$, we consider the sampled directions $D_{y,i,j}$, $D_{v,i,j}$ and $D_{x,i,j}$ defined by

$$D_{y,i,j}(y, v, x) = \nabla_y g_i(x,y) \tag{5.3}$$

$$D_{v,i,j}(y, v, x) = \nabla^2_{yy} g_i(x,y) v + \nabla_y f_j(x,y) \tag{5.4}$$

$$D_{x,i,j}(y, v, x) = \nabla^2_{xy} g_i(x,y) v + \nabla_x f_j(x,y) \ . \tag{5.5}$$

When i and j are randomly sampled uniformly, these directions are unbiased estimators of the true directions D_y, D_v, and D_x. Yet, as in (Nguyen et al., 2017), we use them to recursively build biased estimators of the directions that enable fast convergence.

5.2.3 SRBA: Stochastic Recursive Bilevel Algorithm

In Algorithm 6, we present SRBA, a combination of the idea of recursive gradient coming from (Fang et al., 2018; Nguyen et al., 2022) and the framework proposed in Chapter 4. It relies on a recursive estimation of each direction D_y, D_v, D_x which is updated following the same strategy as SARAH. Let us denote by ρ the step size of the update for the variables y and v, and γ the step size for the update of the variable x. We use the same step size for y and v because the problems of minimizing the inner function g and solving the linear system (5.2) have the same conditioning driven by $\nabla^2_{yy} g$. For simplicity, we denote the joint variable $\mathbf{u} = (y, v, x)$ and the joint directions weighted by the step sizes $\mathbf{\Delta} = (\rho D_y, \rho D_v, \gamma D_x) = (\mathbf{\Delta}_y, \mathbf{\Delta}_v, \mathbf{\Delta}_x)$.

At iteration t, the estimate direction $\mathbf{\Delta}$ is initialized by computing full batch directions:

$$\mathbf{\Delta}^{t,0} = (\rho D_y(\tilde{\mathbf{u}}^t), \rho D_v(\tilde{\mathbf{u}}^t), \gamma D_x(\tilde{\mathbf{u}}^t))$$

Algorithm 6 Stochastic Recursive Bilevel Algorithm

Input: initializations $y_0 \in \mathbb{R}^{d_y}$, $x_0 \in \mathbb{R}^{d_x}$, $v_0 \in \mathbb{R}^{d_y}$, number of iterations T and q, step sizes ρ and γ.
Set $\tilde{\mathbf{u}}^{\mathbf{0}} = (y_0, v_0, x_0)$
for $t = 0, \dots, T-1$ **do**
  Reset $\boldsymbol{\Delta}$: $\boldsymbol{\Delta}^{t,0} = (\rho D_y(\tilde{\mathbf{u}}^t), \rho D_v(\tilde{\mathbf{u}}^t), \gamma D_x(\tilde{\mathbf{u}}^t))$
  Update $\mathbf{u}$: $\mathbf{u}^{t,1} = \Pi(\tilde{\mathbf{u}}^t - \boldsymbol{\Delta}^{\mathbf{t,0}})$,
  for $k = 1, \dots, q-1$ **do**
    Draw $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$
    $\boldsymbol{\Delta}_y^{t,k} = \rho(D_{y,i,j}(\mathbf{u}^{t,k}) - D_{y,i,j}(\mathbf{u}^{t,k-1})) + \boldsymbol{\Delta}_y^{t,k-1}$
    $\boldsymbol{\Delta}_v^{t,k} = \rho(D_{v,i,j}(\mathbf{u}^{t,k}) - D_{v,i,j}(\mathbf{u}^{t,k-1})) + \boldsymbol{\Delta}_v^{t,k-1}$
    $\boldsymbol{\Delta}_x^{t,k} = \gamma(D_{x,i,j}(\mathbf{u}^{t,k}) - D_{x,i,j}(\mathbf{u}^{t,k-1})) + \boldsymbol{\Delta}_x^{t,k-1}$
    Update $\mathbf{u}$: $\mathbf{u}^{t,k+1} = \Pi(\mathbf{u}^{t,k} - \boldsymbol{\Delta}^{t,k})$
  end for
  Set $\tilde{\mathbf{u}}^{t+1} = \mathbf{u}^{t+1,q}$
end for
Return $(\tilde{y}^T, \tilde{v}^T, \tilde{x}^T) = \tilde{\mathbf{u}}^T$

and a first update is performed by moving from $\tilde{\mathbf{u}}^t$ in the direction $-\boldsymbol{\Delta}^{t,0}$. As done by Hu et al. (2022), we project the variable v onto Γ to leverage the boundedness property of v^*. Then, during the kth iteration of an inner loop of size $q-1$, two indices $i \in [n]$ and $j \in [m]$ are sampled and the estimate directions are updated according to Equations (5.6) to (5.8)

$$\boldsymbol{\Delta}_y^{t,k} = \rho(D_{y,i,j}(\mathbf{u}^{t,k}) - D_{y,i,j}(\mathbf{u}^{t,k-1})) + \boldsymbol{\Delta}_y^{t,k-1} \tag{5.6}$$
$$\boldsymbol{\Delta}_v^{t,k} = \rho(D_{v,i,j}(\mathbf{u}^{t,k}) - D_{v,i,j}(\mathbf{u}^{t,k-1})) + \boldsymbol{\Delta}_v^{t,k-1} \tag{5.7}$$
$$\boldsymbol{\Delta}_x^{t,k} = \gamma(D_{x,i,j}(\mathbf{u}^{t,k}) - D_{x,i,j}(\mathbf{u}^{t,k-1})) + \boldsymbol{\Delta}_x^{t,k-1} \tag{5.8}$$

where the sampled directions $D_{y,i,j}$, $D_{v,i,j}$ and $D_{x,i,j}$ are defined by Equations (5.3) to (5.5). Then the joint variable $\mathbf{u}$ is updated by

$$\mathbf{u}^{t,k+1} = \Pi(\mathbf{u}^{t,k} - \boldsymbol{\Delta}^{t,k}) \ . \tag{5.9}$$

Recall that the projection is only performed on the variable v. The other variables y and x remain unchanged after the projection step. At the end of the inner procedure, we set $\tilde{\mathbf{u}}^{t+1} = \mathbf{u}^{t,q}$.

In Algorithm 6, the variables y, v, and x are updated simultaneously rather than alternatively. From a computational perspective, this enables sharing the common computations between the different oracles and doing the update of each variable in parallel. So there is no sub-procedure to approximate the solution of the inner problem and the solution of the linear system.

Note that Yang et al. (2021) propose VRBO, another adaptation of SPIDER/SARAH for bilevel problems. VRBO has a double loop structure where the inner variable is updated by several steps in an inner loop. In this inner loop, the estimates of the gradient of g and the gradient of Φ are also updated using SARAH's update rules. SRBA has a different structure. First, in SRBA, the inner variable y is updated only once between two updates of the outer variable instead of several times. Second, the solution of the linear system evolves following optimization steps, whereas in VRBO, a Neumann approximation is used. Moreover, the algorithm VRBO is analyzed in the case where the functions f and g are general expectations but not in the specific case of empirical risk minimization, as done in Section 5.3. Finally, VRBO requires three more parameters than SRBA: the number of inner steps, the number of terms, and the scaling parameter in the Neumann approximations.

5.3 Theoretical Analysis of SRBA

In this section we provide the theoretical analysis of Algorithm 6 leading to a final sample complexity in $\mathcal{O}\left((n+m)^{\frac{1}{2}}\varepsilon^{-1} \vee (n+m)\right)$. In Definition 5.1, we recall what is an ε-stationary point.

Definition 5.1. Let d a positive integer, $f : \mathbb{R}^d \to \mathbb{R}$ a differentiable function, and $\varepsilon > 0$. We say that a point $x \in \mathbb{R}^d$ is an ε-stationary point of f if $\|\nabla f(x)\|^2 \leq \epsilon$. In a stochastic context, we call ε-stationary point a random variable x such that $\mathbb{E}[\|\nabla f(x)\|^2] \leq \varepsilon$.

In this chapter, the theoretical complexity of the algorithms is given in terms of number of calls to oracle, that is to say, the number of times the quantity

$$[\nabla f_j(x,y), \nabla g_i(x,y), \nabla^2_{yy} g_i(x,y)v, \nabla^2_{xy} g_i(x,y)v] \tag{5.10}$$

is queried for $i \in [n]$, $j \in [m]$, $y \in \mathbb{R}^{d_y}$, $v \in \mathbb{R}^{d_y}$ and $x \in \mathbb{R}^{d_x}$. Note that in practice, although the second-order derivatives of the inner functions $\nabla^2_{yy} g_i(x,y) \in \mathbb{R}^{p\times p}$ and $\nabla^2_{xy} g_i(x,y) \in \mathbb{R}^{d\times p}$ are involved, they are never computed or stored explicitly. We rather work with Hessian-vector products $\nabla^2_{yy} g_i(x,y)v \in \mathbb{R}^{d_y}$ and Jacobian-vector products $\nabla^2_{xy} g_i(x,y)v \in \mathbb{R}^{d_x}$ which can be computed efficiently thanks to automatic differentiation with a computational cost similar to the cost of computing the gradients $\nabla_y g_i(x,y)$ and $\nabla_x g_i(x,y)$ as explained in Section 2.3 (Pearlmutter, 1994). The cost of one query (5.10) is, therefore, of the same order of magnitude as that of computing one stochastic gradient.

5.3.1 Mean Squared Error of the Estimated Directions

A strength of our method is its simple expression of the estimation error of the directions coming from the bias-variance decomposition provided by Nguyen et al. (2017). Let us denote the estimate directions $D_y^{t,k} = \boldsymbol{\Delta}_y^{t,k}/\rho$, $D_v^{t,k} = \boldsymbol{\Delta}_v^{t,k}/\rho$ and $D_x^{t,k} = \boldsymbol{\Delta}_x^{t,k}/\gamma$. In the remainder of the chapter, a quantity denoted $A_\bullet$ refers to A_y, A_v or A_x, depending on the context. We introduce the residuals

$$S_\bullet^{t,k} = \sum_{r=1}^{k} \mathbb{E}[\|D_\bullet(\mathbf{u}^{t,r}) - D_\bullet(\mathbf{u}^{t,r-1})\|^2],$$
$$\tilde{S}_\bullet^{t,k} = \sum_{r=1}^{k} \mathbb{E}[\|D_\bullet^{t,r} - D_\bullet^{t,r-1}\|^2] \ .$$

We provide a link between the mean squared error $\mathbb{E}[\|D_\bullet^{t,k} - D_\bullet(\mathbf{u}^{t,k})\|^2]$ and the residuals.

Proposition 5.4 (MSE of the estimate directions)**.** *For any $t \geq 0$ and $k \in \{1,\dots,q-1\}$, the estimate $D_\bullet^{t,k}$ of the direction $D_\bullet(\mathbf{u}^{t,k})$ satisfies*

$$\mathbb{E}[\|D_\bullet^{t,k} - D_\bullet(\mathbf{u}^{t,k})\|^2] = \tilde{S}_\bullet^{t,k} - S_\bullet^{t,k} \ .$$

Proof. Let $t > 0$ and $k \in [q-1]$.

For $k = 0$, we directly have $\mathbb{E}[\|D_\bullet^{t,k} - D_\bullet(\mathbf{u}^{t,k})\|^2] = 0$.

For $k \geq 1$ and $r \in \{1,\dots,k\}$, the bias/variance decomposition of $D_\bullet^{t,r}$ reads

$$\begin{aligned}
\mathbb{E}_{t,r}[\|D_\bullet^{t,r} - D_\bullet(\mathbf{u}^{t,r})\|^2] &= \mathbb{E}_{t,r}[\|D_\bullet^{t,r} - D_\bullet^{t,r-1} + D_\bullet(\mathbf{u}^{t,r-1}) - D_\bullet(\mathbf{u}^{t,r})\|^2] \\
&\quad + \|D_\bullet(\mathbf{u}^{t,r}) + D_\bullet(\mathbf{u}^{t,r-1}) - D_\bullet^{t,r-1} - D_\bullet(\mathbf{u}^{t,r})\|^2 \\
&= \mathbb{E}_{t,r}[\|D_\bullet^{t,r} - D_\bullet^{t,r-1} - (D_\bullet(\mathbf{u}^{t,r-1}) - D_\bullet(\mathbf{u}^{t,r}))\|^2] \\
&\quad + \|D_\bullet^{t,r-1} - D_\bullet(\mathbf{u}^{t,r-1})\|^2
\end{aligned}$$

The term $\mathbb{E}_{t,r}[\|D_\bullet^{t,r} - D_\bullet^{t,r-1} - (D_\bullet(\mathbf{u}^{t,r-1}) - D_\bullet(\mathbf{u}^{t,r}))\|^2]$ is the variance of $D_\bullet^{t,r} - D_\bullet^{t,r-1}$, and then can written as

$$\mathbb{E}_{t,r}[\|D_\bullet^{t,r} - D_\bullet^{t,r-1} - (D_\bullet(\mathbf{u}^{t,r-1}) - D_\bullet(\mathbf{u}^{t,r}))\|^2] = \mathbb{E}_{t,r}[\|D_\bullet^{t,r} - D_\bullet^{t,r-1}\|^2] - \|D_\bullet(\mathbf{u}^{t,r}) - D_\bullet(\mathbf{u}^{t,r-1})\|^2$$

Plugging this in the previous inequality and taking the total expectation leads to

$$\mathbb{E}[\|D_\bullet^{t,r} - D_\bullet(\mathbf{u}^{t,r})\|^2] = \mathbb{E}[\|D_\bullet^{t,r} - D_\bullet^{t,r-1}\|^2] - \mathbb{E}[\|D_\bullet(\mathbf{u}^{t,r}) - D_\bullet(\mathbf{u}^{t,r-1})\|^2] + \mathbb{E}[\|D_\bullet^{t,r-1} - D_\bullet^{t,r-1}(\mathbf{u}^{t,r-1})\|^2]$$

Summing for $r \in \{1,\dots,k\}$ and telescoping gives the final result (taking into account that we have $D_\bullet^{t,0} = D_\bullet(\mathbf{u}^{t,0})$):

$$\mathbb{E}[\|D_\bullet^{t,k} - D_\bullet(\mathbf{u}^{t,k})\|^2] = \sum_{r=1}^{k} \mathbb{E}[\|D_\bullet^{t,r} - D_\bullet^{t,r-1}\|^2] - \sum_{r=1}^{k} \mathbb{E}[\|D_\bullet(\mathbf{u}^{t,r}) - D_\bullet(\mathbf{u}^{t,r-1})\|^2] \ .$$

□

The above error has two components: the accumulation of the difference between two successive full batch directions and the accumulation of the difference between two successive estimate directions.

5.3.2 Fundamental Lemmas

We establish descent lemmas which are key ingredients to get the final convergence result. Lemma 5.1 characterizes the dynamic of $\mathbf{u}$ on the inner problem. To do so, we define the function ϕ_y as

$$\phi_y(x,y) = g(x,y) - g(x,y^*(x)) \ .$$

In the bilevel literature, a direct control on the distance to optimum $\delta_y^{t,k} \triangleq \mathbb{E}[\|y^{t,k} - y^*(x^{t,k})\|^2]$ is established. Here, the biased nature of the estimate direction $D_y^{t,k}$ makes it hard to upper bound appropriately the scalar product $\langle D_y(\mathbf{u}^{t,k}) - D_y^{t,k}, y^{t,k} - y^*(x^{t,k})\rangle$. Therefore, we rather consider $\phi_y^{t,k}$. By combining the smoothness property of ϕ_y and the bias-variance decomposition provided in Proposition 5.4, we can show some descent property on the sequence $\phi_y^{t,k}$ defined by $\phi_y^{t,k} = \mathbb{E}[\phi_y(y^{t,k}, x^{t,k})]$. Before stating Lemma 5.1, let us define $\mathcal{G}_v^{t,k} = \frac{1}{\rho}\left(v^{t,k} - \Pi_\Gamma(v^{t,k} - \rho D_v^{t,k})\right)$ so that $v^{t,k+1} = v^{t,k} - \rho\mathcal{G}_v^{t,k}$. This is the actual update direction of v. If there were no projections, we would have $\mathcal{G}_v^{t,k} = D_v^{t,k}$. Hence, it acts as a surrogate of $D_v^{t,k}$ in our analysis. We also define

$$V_y^{t,k} = \mathbb{E}[\|D_y^{t,k}\|^2], \quad V_v^{t,k} = \mathbb{E}[\|\mathcal{G}_v^{t,k}\|^2], \quad V_x^{t,k} = \mathbb{E}[\|D_x^{t,k}\|^2]$$

the variances and their respective sums over the inner loop

$$\mathcal{V}_y^{t,k} = \sum_{r=1}^{k} \mathbb{E}[\|D_y^{t,r-1}\|^2], \quad \mathcal{V}_v^{t,k} = \sum_{r=1}^{k} \mathbb{E}[\|\mathcal{G}_v^{t,r-1}\|^2], \quad \mathcal{V}_x^{t,k} = \sum_{r=1}^{k} \mathbb{E}[\|D_x^{t,r-1}\|^2] \ .$$

Lemma 5.1 (Descent on the inner level)**.** *Assume that the step sizes ρ and γ verify $\gamma \leq C_y\rho$ for some positive constant C_y specified in the proof. Then it holds*

$$\phi_y^{t,k+1} \leq \left(1 - \frac{\mu_g}{2}\rho\right)\phi_y^{t,k} - \frac{\rho}{2}(1 - \Lambda_y\rho)\, V_y^{t,k} + \rho^3\beta_{yy}\mathcal{V}_y^{t,k} + \gamma^2\rho\beta_{yv}\mathcal{V}_v^{t,k} + \gamma^2\rho\beta_{yx}\mathcal{V}_x^{t,k} + \frac{\Lambda_y}{2}\gamma^2 V_x^{t,k} + \frac{\gamma^2}{\rho}\overline{\beta}_{yx}\mathbb{E}[\|D_x(\mathbf{u}^{t,k})\|^2] \tag{5.11}$$

for some positive constants $\Lambda_y, \beta_{yy}, \beta_{yx}$ and $\overline{\beta}_{yx}$ that are specified in the proof.

In (5.11) we recover a decrease of $\phi_y^{t,k}$ by a factor $(1 - \rho\mu_g)$. But the outer variable's movement and the noise make appear $D_x(\mathbf{u}^{t,k})$ and the variance hindering the convergence of y towards $y^*(x)$.

Proof. In subsection B.1.2, we show the Γ_y-smoothness of ϕ_y. This enables us to write:

$$\phi_y(y^{t,k+1}, x^{t,k+1}) \leq \phi_y(y^{t,k}, x^{t,k}) - \rho\langle D_y^{t,k}, \nabla_y g(y^{t,k}, x^{t,k})\rangle + \frac{\Lambda_y}{2}\rho^2\|D_y^{t,k}\|^2 - \gamma\langle D_x^{t,k}, \nabla_x g(y^{t,k}, x^{t,k}) - \nabla_x g(y^*(x^{t,k}), x^{t,k})\rangle + \frac{\Lambda_y}{2}\gamma^2\|D_x^{t,k}\|^2 \ . \tag{5.12}$$

Using the equality $\langle a, b\rangle = \frac{1}{2}(\|a\|^2 + \|b\|^2 - \|a-b\|^2)$, we get

$$-\langle D_y^{t,k}, \nabla_y g(y^{t,k}, x^{t,k})\rangle + \frac{\Lambda_y}{2}\rho\|D_y^{t,k}\|^2 = \frac{1}{2}(\|D_y^{t,k} - \nabla_y g(y^{t,k}, x^{t,k})\|^2 - \|\nabla_y g(y^{t,k}, x^{t,k})\|^2 - (1-\Lambda_y\rho)\,\|D_y^{t,k}\|^2) \ . \tag{5.13}$$

Plugging Equation (5.13) into Equation (5.12) and tacking the expectation conditionally to the past iterates yields

$$\begin{aligned}\mathbb{E}_{t,k}[\phi_y^{t,k+1}] \leq \phi_y^{t,k} &+ \frac{\rho}{2}\mathbb{E}_{t,k}[\|D_y^{t,k} - \nabla_y g(y^{t,k}, x^{t,k})\|^2] \\ &- \frac{\rho}{2}\|\nabla_y g(y^{t,k}, x^{t,k})\|^2 - \frac{\rho}{2}(1-\Lambda_y\rho)\,\mathbb{E}_{t,k}[\|D_y^{t,k}\|^2] \\ &- \gamma\langle \mathbb{E}_{t,k}[D_x^{t,k}], \nabla_x g(y^{t,k}, x^{t,k}) - \nabla_x g(y^*(x^{t,k}), x^{t,k})\rangle + \frac{\Lambda_y}{2}\gamma^2\mathbb{E}_{t,k}[\|D_x^{t,k}\|^2] \ .\end{aligned} \tag{5.14}$$

From Young inequality, we have for any $c > 0$

$$\begin{aligned}\langle \mathbb{E}_{t,k}[D_x^{t,k}], \nabla_x g(y^{t,k}, x^{t,k}) - \nabla_x g(y^*(x^{t,k}), x^{t,k})\rangle \leq \frac{1}{2c}\|\mathbb{E}_{t,k}[D_x^{t,k}]\|^2 \\ + \frac{c}{2}\|\nabla_x g(y^{t,k}, x^{t,k}) - \nabla_x g(y^*(x^{t,k}), x^{t,k})\|^2\end{aligned} \tag{5.15}$$

The smoothness of g and strong convexity give us

$$\|\nabla_x g(y^{t,k}, x^{t,k}) - \nabla_x g(y^*(x^{t,k}), x^{t,k})\|^2 \leq L_{g,1}\|y^{t,k} - y^*(x^{t,k})\|^2 \leq \frac{2L_{g,1}}{\mu_g}\phi_y(y^{t,k}, x^{t,k}) \tag{5.16}$$

Let us denote $L' = \frac{L_{g,1}}{\mu_g}$. Plugging inequalities (5.15) and (5.16) into Equation (5.14) yields

$$\begin{aligned}\mathbb{E}_{t,k}[\phi_y(y^{t,k+1}, x^{t,k+1})] \leq (1 + cL'\gamma)\phi_y(y^{t,k+1}, x^{t,k+1}) &- \frac{\rho}{2}\mathbb{E}_{t,k}[\|\nabla_y g(y^{t,k}, x^{t,k})\|^2] \\ &+ \frac{\rho}{2}\mathbb{E}_{t,k}[\|D_y^{t,k} - \nabla_y g(y^{t,k}, x^{t,k})\|^2] - \frac{\rho}{2}(1-\Lambda_y\rho)\,\mathbb{E}_{t,k}[\|D_y^{t,k}\|^2] \\ &+ \frac{\gamma}{2c}\|\mathbb{E}_{t,k}[D_x^{t,k}]\|^2 + \frac{\Lambda_y}{2}\gamma^2\mathbb{E}_{t,k}[\|D_x^{t,k}\|^2]\end{aligned} \tag{5.17}$$

From Lemma B.2, we have

$$\mathbb{E}[\|D_y^{t,k} - \nabla_y g(y^{t,k}, x^{t,k})\|^2] \leq \sum_{r=1}^{k} L_{g,1}(\rho^2\mathbb{E}[\|D_y^{t,r-1}\|^2] + \gamma^2\mathbb{E}[\|D_y^{t,r-1}\|^2]) \ .$$

Taking the total expectation and plugging the previous inequality into Equation (5.17) yields

$$\begin{aligned}\phi_y^{t,k+1} \leq (1 + cL'\gamma)\phi^{t,k} &+ \frac{L_{g,1}}{2}\sum_{r=1}^{k}(\rho^3\mathbb{E}[\|D_y^{t,r-1}\|^2] + \gamma^2\rho\mathbb{E}[\|D_x^{t,r-1}\|^2]) \\ &- \frac{\rho}{2}\mathbb{E}[\|\nabla_y g(y^{t,k}, x^{t,k})\|^2] - \frac{\rho}{2}(1-\Lambda_y\rho)\,\mathbb{E}[\|D_y^{t,k}\|^2] \\ &+ \frac{\gamma}{2c}\mathbb{E}[\|\mathbb{E}[D_x^{t,k}]\|^2] + \frac{\Lambda_y}{2}\gamma^2\mathbb{E}[\|D_x^{t,k}\|^2]\end{aligned} \tag{5.18}$$

Since g is μ_g-strongly convex with respect to z, Polyak-Łojasiewicz inequality holds:

$$\|\nabla_y g(y^{t,k}, x^{t,k})\|^2 \geq 2\mu_g\phi_y(y^{t,k}, x^{t,k})$$

As a consequence, Equation (5.18) becomes

$$\begin{aligned}\phi_y^{t,k+1} \leq (1 + cL'\gamma - \mu_g\rho)\,\phi^{t,k} &+ \frac{L_{g,1}}{2}\sum_{r=1}^{k}(\rho^3\mathbb{E}[\|D_y^{t,r-1}\|^2] + \gamma^2\rho\mathbb{E}[\|D_x^{t,r-1}\|^2]) \\ &- \frac{\rho}{2}(1-\Lambda_y\rho)\,\mathbb{E}[\|D_y^{t,k}\|^2] + \frac{\gamma}{2c}\mathbb{E}[\|\mathbb{E}[D_x^{t,k}]\|^2] + \frac{\Lambda_y}{2}\gamma^2\mathbb{E}[\|D_x^{t,k}\|^2]\end{aligned}$$

Taking $c = \frac{\mu_g \rho}{2L'\gamma}$ yields

$$\phi_y^{t,k+1} \leq \left(1 - \frac{\mu_g}{2}\rho\right)\phi^{t,k} + \frac{L_{g,1}}{2}\sum_{r=1}^{k}(\rho^3\mathbb{E}[\|D_y^{t,r-1}\|^2] + \gamma^2\rho\mathbb{E}[\|D_x^{t,r-1}\|^2]) - \frac{\rho}{2}\left(1 - \Lambda_y\rho\right)\mathbb{E}[\|D_y^{t,k}\|^2] + \frac{L'}{\mu_g}\frac{\gamma^2}{\rho}\mathbb{E}[\|\mathbb{E}[D_x^{t,k}]\|^2] + \frac{\Lambda_y}{2}\gamma^2\mathbb{E}[\|D_x^{t,k}\|^2]$$

For the term $\mathbb{E}[\|\mathbb{E}_{t,k}[D_y^{t,k}]\|^2]$, we have

$$\begin{aligned}\mathbb{E}[\|\mathbb{E}_{t,k}[D_x^{t,k}]\|^2] &= \mathbb{E}[\|D_x(y^{t,k}, v^{t,k}, x^{t,k}) - D_x(y^{t,k-1}, v^{t,k-1}, x^{t,k-1}) + D_x^{t,k-1}\|^2] \\ &= \mathbb{E}[\|D_x(y^{t,k}, v^{t,k}, x^{t,k}) - D_x(y^{t,k-1}, v^{t,k-1}, x^{t,k-1}) - \mathbb{E}[D_x^{t,k-1}]\|^2] \\ &\quad + \mathbb{E}[\|D_x^{t,k-1} - \mathbb{E}[D_x^{t,k-1}]\|^2] \\ &= \mathbb{E}[\|D_x(y^{t,k}, v^{t,k}, x^{t,k})\|^2] \\ &\quad + \mathbb{E}[\|D_x^{t,k-1} - D_x(y^{t,k-1}, v^{t,k-1}, x^{t,k-1})\|^2] \ .\end{aligned} \tag{5.19}$$

Using Lemma B.2, we get

$$\begin{aligned}\mathbb{E}[\|D_x^{t,k-1} - D_x(\mathbf{u}^{t,k-1})\|^2] \leq 4\rho^2\left((L_{g,2}R)^2 + (L_{f,1})^2\right)\sum_{r=1}^{k-1}\mathbb{E}[\|D_y^{t,r-1}\|^2] \\ + 4\rho^2(L_{g,1})^2\sum_{r=1}^{k-1}\mathbb{E}[\|\mathcal{G}_v^{t,r-1}\|^2] \\ + 4\gamma^2\left((L_{g,2}R)^2 + (L_{f,1})^2\right)\sum_{r=1}^{k-1}\mathbb{E}[\|D_x^{t,r-1}\|^2] \ .\end{aligned}$$

Putting all together yields

$$\begin{aligned}\phi_y^{t,k+1} \leq \left(1 - \frac{\mu_g}{2}\rho\right)\phi^{t,k} - \frac{\rho}{2}\left(1 - \Lambda_y\rho\right)\mathbb{E}[\|D_y^{t,k}\|^2] + \frac{\Lambda_y}{2}\gamma^2\mathbb{E}[\|D_x^{t,k}\|^2] \\ + \frac{L'}{\mu_g}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x^{t,k}(\mathbf{u}^{t,k})\|^2] + 4(L_{g,1})^2\frac{L'}{\mu_g}\gamma^2\rho\sum_{r=1}^{k}\mathbb{E}[\|\mathcal{G}_v^{t,r-1}\|^2] \\ + \rho\left[\rho^2\frac{L_{g,1}}{2} + \frac{4(L_{g,2}R)^2L'}{\mu_g}\gamma^2 + \frac{4(L_{f,1})^2L'}{\mu_g}\gamma^2\right]\sum_{r=1}^{k}\mathbb{E}[\|D_y^{t,r-1}\|^2] \\ + \gamma^2\left[\rho\frac{L_{g,1}}{2} + 4(L_{g,2}R)^2\frac{L'}{\mu_g}\frac{\gamma^2}{\rho} + 4(L_{f,1})^2\frac{L'}{\mu_g}\frac{\gamma^2}{\rho}\right]\sum_{r=1}^{k}\mathbb{E}[\|D_x^{t,r-1}\|^2]\end{aligned} \tag{5.20}$$

By assumption, $\gamma \leq C_y\rho$, with $C_y = \sqrt{\frac{\mu_g L_{g,1}}{8L'((L_{g,2}R)^2 + (L_{f,1})^2)}}$ therefore

$$\begin{aligned}\phi_y^{t,k+1} &\leq \left(1 - \frac{\mu_g}{2}\rho\right)\phi_y^{t,k} - \frac{\rho}{2}\left(1 - \Lambda_y\rho\right)\mathbb{E}[\|D_y^{t,k}\|^2] + \frac{\Lambda_y}{2}\gamma^2\mathbb{E}[\|D_x^{t,k}\|^2] \\ &\quad + \frac{L'}{\mu_g}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x^{t,k}(\mathbf{u}^{t,k})\|^2] + \rho^3 L_{g,1}\sum_{r=1}^{k}\mathbb{E}[\|D_y^{t,r-1}\|^2] \\ &\quad + 4(L_{g,1})^2\frac{L'}{\mu_g}\gamma^2\rho\sum_{r=1}^{k}\mathbb{E}[\|\mathcal{G}_v^{t,r-1}\|^2] + \gamma^2\rho L_{g,1}\sum_{r=1}^{k}\mathbb{E}[\|D_x^{t,r-1}\|^2] \\ &\leq \left(1 - \frac{\mu_g}{2}\rho\right)\phi^{t,k} - \frac{\rho}{2}\left(1 - \Lambda_y\rho\right)V_y^{t,k} + \frac{\Lambda_y}{2}\gamma^2 V_x^{t,k} + \overline{\beta}_{yx}\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x^{t,k}(\mathbf{u}^{t,k})\|^2] \\ &\quad + \rho^3\beta_{yy}\mathcal{V}_y^{t,k} + \gamma^2\rho\beta_{yv}\mathcal{V}_v^{t,k} + \gamma^2\rho\beta_{yx}\mathcal{V}_x^{t,k}\end{aligned}$$

with $\beta_{yy} = L_{g,1}$, $\beta_{yv} = \frac{4(L_{g,1})^2L'}{\mu_g}$, $\beta_{yx} = L_{g,1}$ and $\overline{\beta}_{yx} = \frac{L'}{\mu_g}$. □

For the variable v, the quantity we consider is

$$\phi_v(v,x) \;=\; \Psi(y^*(x),v,x) - \Psi(y^*(x),v^*(x),x)$$

where $\Psi(y,v,x)$ is defined as

$$\Psi(y,v,x) = \frac{1}{2}v^\top \nabla^2_{yy} g(x,y)v + \nabla_y f(x,y)^\top v \;\;.$$

The intuition behind considering this quantity is that solving the linear system (5.2) is equivalent to minimizing over v the function $\Psi(y^*(x),v,x)$.

Lemma 5.2. *Assume that the step sizes ρ and γ verify $\rho \leq B_v$ and $\gamma \;\leq\; C_v\rho$ for some positive constants B_v and C_v specified in the appendix. Then it holds*

$$\begin{aligned}\phi_v^{t,k+1} \leq \left(1 - \frac{\rho\mu_g}{16}\right)\phi_v^{t,k} - \tilde{\beta}_{vv}\rho V_v^t + \rho^3\beta_{vy}\mathcal{V}_y^{t,k} + 2\rho^3\beta_{vv}\mathcal{V}_v^{t,k} + \gamma^2\rho\beta_{vx}\mathcal{V}_x^{t,k} \\ + \rho\alpha_{vy}\phi_y^{t,k} + \frac{\Lambda_v}{2}\gamma^2\mathbb{E}[\|D_x^{t,k}\|^2] + \frac{\gamma^2}{\rho}\overline{\beta}_{vx}\mathbb{E}[\|D_x(\mathbf{u}^{t,k})\|^2]\end{aligned}$$

for some positive constants $\Lambda_v, \beta_{vy}, \beta_{vx}, \tilde{\beta}_{vv}$ and $\overline{\beta}_{vx}$ that are specified in the appendix.

Lemma 5.2 is similar to Lemma 5.1 with a term in $\phi_y^{t,k}$ taking into account the error of $y^*(x)$'s approximation. The proof of this lemma is provided in subsection B.1.3. It harnesses the generalization of Polyak-Łojasiewicz inequality for composite functions introduced by Karimi et al. (2016).

The following lemma is a consequence of the smoothness of Φ. Let us denote the expected values $\Phi^{t,k} = \mathbb{E}[\Phi(x^{t,k})]$ and expected gradient $g^{t,k} \;=\; \mathbb{E}[\|\nabla\Phi(x^{t,k})\|^2]$.

Lemma 5.3. *There exist constants $\beta_{\Phi y}, \beta_{\Phi v}, \beta_{\Phi x} > 0$ such that*

$$\begin{aligned}\Phi^{t,k+1} \leq \Phi^{t,k} - \frac{\gamma}{2}g^{t,k} + \gamma\frac{2L_x^2}{\mu_g}(\phi_y^{t,k} + \phi_v^{t,k}) + \gamma\rho^2\beta_{\Phi y}\mathcal{V}_y^{t,k} \\ + \gamma\rho^2\beta_{\Phi v}\mathcal{V}_v^{t,k} + \gamma^3\beta_{\Phi x}\mathcal{V}_x^{t,k} - \frac{\gamma}{2}\left(1 - L_\Phi\gamma\right)V_x^{t,k} \;\;.\end{aligned}$$

This lemma shows that the control of the approximation error $\phi_\bullet$ (Lemma 5.1 and Lemma 5.2) and the sum of variances $\mathcal{V}_\bullet$ is crucial to get a decrease of $\mathbb{E}[\Phi(x^{t,k})]$.

Proof. The smoothness of Φ (Proposition 3.2) gives us

$$\Phi(x^{t,k+1}) \leq \Phi(x^{t,k}) - \gamma\langle\nabla\Phi(x^{t,k}), D_x^{t,k}\rangle + \gamma^2\frac{L_\Phi}{2}\|D_x^{t,k}\|^2 \;\;.$$

Then, we use the identity $\langle a,b\rangle = \frac{1}{2}(\|a\|^2 + \|b\|^2 - \|a-b\|^2)$ to get

$$\begin{aligned}\Phi(x^{t,k+1}) &\leq \Phi(x^{t,k}) - \frac{\gamma}{2}\|\nabla\Phi(x^{t,k})\|^2 - \frac{\gamma}{2}\|D_x^{t,k}\|^2 + \frac{\gamma}{2}\|\nabla\Phi(x^{t,k}) - D_x^{t,k}\|^2 + \gamma^2\frac{L_\Phi}{2}\|D_x^{t,k}\|^2 \\ &\leq \Phi(x^{t,k}) - \frac{\gamma}{2}\|\nabla\Phi(x^{t,k})\|^2 - \frac{\gamma}{2}\|D_x^{t,k}\|^2 + \gamma\|\nabla\Phi(x^{t,k}) - D_x(\mathbf{u}^{t,k})\|^2 \\ &\quad + \gamma\|D_x(\mathbf{u}^{t,k}) - D_x^{t,k}\|^2 + \gamma^2\frac{L_\Phi}{2}\|D_x^{t,k}\|^2 \;\;.\end{aligned}$$

Then taking the expectation gives and using Proposition 5.2 yields

$$\begin{aligned}\Phi^{t,k+1} &\leq \Phi^{t,k} - \frac{\gamma}{2}g^{t,k} + \gamma\mathbb{E}[\|\nabla\Phi(x^{t,k}) - D_x(\mathbf{u}^{t,k})\|^2] \\ &\quad + \gamma\mathbb{E}[\|D_x(\mathbf{u}^{t,k}) - D_x^{t,k}\|^2] - \frac{\gamma}{2}\left(1 - L_\Phi\gamma\right)\mathbb{E}[\|D_x^{t,k}\|^2] \\ &\leq \Phi^{t,k} - \frac{\gamma}{2}g^{t,k} + \gamma L_x^2(\mathbb{E}[\|y^{t,k} - y^*(x^{t,k})\|^2] + \mathbb{E}[\|v^{t,k} - v^*(x^{t,k})\|^2]) \\ &\quad + \gamma\mathbb{E}[\|D_x(\mathbf{u}^{t,k}) - D_x^{t,k}\|^2] - \frac{\gamma}{2}\left(1 - L_\Phi\gamma\right)\mathbb{E}[\|D_x^{t,k}\|^2] \;\;.\end{aligned}$$

The μ_g-strong convexity of g ensures that $\|y - y^*(x)\|^2 \leq \frac{2}{\mu_g}\phi_y(y,x)$ and $\|v - v^*(x)\|^2 \leq \frac{2}{\mu_g}\phi_v(v,x)$. As a consequence

$$\Phi^{t,k+1} \leq \Phi^{t,k} - \frac{\gamma}{2}g^{t,k} + \gamma\frac{2L_x^2}{\mu_g}(\phi_y^{t,k} + \phi_v^{t,k}) + \gamma\mathbb{E}[\|D_x(y^{t,k}, v^{t,k}, x^{t,k}) - D_x^{t,k}\|^2] - \frac{\gamma}{2}\left(1 - L_\Phi\gamma\right)\mathbb{E}[\|D_x^{t,k}\|^2] \ .$$

From Lemma B.2, we have

$$\mathbb{E}[\|D_x^{t,k} - D_x(\mathbf{u}^{t,k})\|^2] \leq 4\rho^2\left((L_{g,2}R)^2 + (L_{f,1})^2\right)\sum_{r=1}^{k}\mathbb{E}[\|D_y^{t,r-1}\|^2] + 4\rho^2(L_{g,1})^2\sum_{r=1}^{k}\mathbb{E}[\|\mathcal{G}_v^{t,r-1}\|^2] + 4\gamma^2\left((L_{g,2}R)^2 + (L_{f,1})^2\right)\sum_{r=1}^{k}\mathbb{E}[\|D_x^{t,r-1}\|^2] \ .$$

As a consequence

$$\begin{aligned}\Phi^{t,k+1} &\leq \Phi^{t,k} - \frac{\gamma}{2}g^{t,k} + \gamma\frac{2L_x^2}{\mu_g}(\phi_y^{t,k} + \phi_v^{t,k}) \\ &\quad + 4\gamma\rho^2\left((L_{g,2}R)^2 + 2(L_{f,1})^2\right)\sum_{r=1}^{k}\mathbb{E}[\|D_y^{t,r-1}\|^2] + 4\gamma\rho^2(L_1^G)^2\sum_{r=1}^{k}\mathbb{E}[\|\mathcal{G}_v^{t,r-1}\|^2] \\ &\quad + 4\gamma^3\left((L_{g,2}R)^2 + 2(L_{f,1})^2\right)\sum_{r=1}^{k}\mathbb{E}[\|D_x^{t,r-1}\|^2] - \frac{\gamma}{2}\left(1 - L_\Phi\gamma\right)\mathbb{E}[\|D_x^{t,k}\|^2] \\ &\leq \Phi^{t,k} - \frac{\gamma}{2}g^{t,k} + \gamma\frac{2L_x^2}{\mu_g}(\phi_y^{t,k} + \phi_v^{t,k}) + \gamma\rho^2\beta_{\Phi y}\mathcal{V}_y^{t,k} + \gamma\rho^2\beta_{\Phi v}\mathcal{V}_v^{t,k} \\ &\quad + \gamma^3\beta_{\Phi x}\mathcal{V}_x^{t,k} - \frac{\gamma}{2}\left(1 - L_\Phi\gamma\right)\mathbb{E}[\|D_x^{t,k}\|^2]\end{aligned}$$

with $\beta_{\Phi y} = 4\left((L_{g,2}R)^2 + 2(L_{f,1})^2\right)$, $\beta_{\Phi v} = 4(L_1^G)^2$ and $\beta_{\Phi x} = 4\left((L_{g,2}R)^2 + 2(L_{f,1})^2\right)$. □

5.3.3 Complexity Analysis of SRBA

In Theorem 5.1, we provide the convergence rate of SRBA towards a stationary point.

Theorem 5.1. *Assume that Assumptions 5.1 and 5.2 hold. Assume that the step sizes verify $\rho \leq \overline{\rho}$ and $\gamma \leq \min(\overline{\gamma}, \xi\rho)$ for some constants ξ, $\overline{\rho}$ and $\overline{\gamma}$ specified in appendix. Then it holds*

$$\frac{1}{Tq}\sum_{t=0}^{T-1}\sum_{k=0}^{q-1}\mathbb{E}[\|\nabla\Phi(x^{t,k})\|^2] = \mathcal{O}\left(\frac{1}{qT\gamma}\right)$$

where $\mathcal{O}$ hides regularity constants that are independent from n and m.

Proof sketch. The detailed proof is provided in subsection B.1.4.

The proof combines classical proof techniques from the bilevel literature and elements from SARAH's analysis (Nguyen et al., 2017, 2022). We introduce the Lyapunov function

$$\mathcal{L}^{t,k} = \Phi^{t,k} + \psi_y\phi_y^{t,k} + \psi_v\phi_v^{t,k}$$

where ψ_y and ψ_v are non-negative constants.

By combining Lemmas 5.1 to 5.3 we get

$$\begin{aligned}\mathcal{L}^{t,k+1} - \mathcal{L}^{t,k} \leq & -\frac{\gamma}{2} g^{t,k} + A\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(\mathbf{u}^{t,k})\|^2] + (B_1\gamma - B_2\rho + B_3\rho)\,\phi_y^{t,k} \\ & + (C_1\gamma - C_2\rho)\,\phi_v^{t,k} + \left(D_1\rho^2 - D_2\rho\right) V_y^{t,k} - E_1\rho V_v^{t,k} \\ & + \left(F_1\gamma^2 - F_2\gamma\right) V_x^{t,k} + \left(G_1\rho\gamma^2 + G_2\rho^3 + G_3\rho^3\right) \mathcal{V}_y^{t,k} \\ & + \left(H_1\rho\gamma^2 + H_2\gamma^2\rho + H_3\rho^3\right) \mathcal{V}_v^{t,k} + \left(I_1\gamma^3 + I_2\gamma^2\rho + I_3\rho^3\right) \mathcal{V}_x^{t,k} \ ,\end{aligned} \tag{5.21}$$

where the constants A, B_1, B_2, B_3, C_1, C_2, D_1, D_2, E_1, F_1, F_2, G_1, G_2, G_3, H_1, H_2, H_3, I_1, I_2 and I_3 are independent from the step sizes and q.

The term $\mathbb{E}[\|D_x(\mathbf{u}^{t,k})\|^2]$ can be bounded by using Proposition 5.2:

$$\mathbb{E}[\|D_x(\mathbf{u}^{t,k})\|^2] \leq 2g^{t,k} + \frac{4}{\mu_g}(\phi_y^{t,k} + \phi_v^{t,k})$$

If we inject the previous equation in Equation (5.21), then by summing a telescoping, we get

$$\begin{aligned}\mathcal{L}^{t,q} - \mathcal{L}^{t,0} \leq & -\left(\frac{\gamma}{2} - J\frac{\gamma^2}{\rho}\right)\sum_{k=0}^{q-1} g^{t,k} + \left(B_1\gamma - B_2\rho + B_3\rho + B_4\frac{\gamma^2}{\rho}\right)\sum_{k=0}^{q-1}\phi_y^{t,k} \\ & + \left(C_1\gamma - C_2\rho + C_3\frac{\gamma^2}{\rho}\right)\sum_{k=0}^{q-1}\phi_v^{t,k} + (D_1\rho^2 - D_2\rho)\sum_{k=0}^{q-1} V_y^{t,k} \\ & - E_1\rho\sum_{k=0}^{q-1} V_v^{t,k} + \left(F_1\gamma^2 - F_2\gamma\right)\sum_{k=0}^{q-1} V_x^{t,k} + \left(G_1\rho\gamma^2 + G_2\rho^3 + G_3\rho^3\right)\sum_{k=0}^{q-1}\mathcal{V}_y^{t,k} \\ & + \left(H_1\rho\gamma^2 + H_2\gamma^2\rho + H_3\rho^3\right)\sum_{k=0}^{q-1}\mathcal{V}_v^{t,k} \left(I_1\gamma^3 + I_2\gamma^2\rho + I_3\rho^3\right)\sum_{k=0}^{q-1}\mathcal{V}_x^{t,k} \ ,\end{aligned}$$

Using the inequality

$$\sum_{k=0}^{q-1}\mathcal{V}_\bullet^{t,k} \leq q\sum_{k=1}^{q-1}\mathbb{E}[\|D_\bullet^{t,k-1}\|^2] \ ,$$

we get

$$\begin{aligned}\mathcal{L}^{t,q} - \mathcal{L}^{t,0} \leq & -\left(\frac{\gamma}{2} - J'\frac{\gamma^2}{\rho}\right)\sum_{k=0}^{q-1} g^{t,k} \\ & + \left(B_1\gamma - B_2\rho + B_3\rho + B_4\frac{\gamma^2}{\rho}\right)\sum_{k=0}^{q-1}\phi_y^{t,k} + \left(C_1\gamma - C_2\rho + C_3\frac{\gamma^2}{\rho}\right)\sum_{k=0}^{q-1}\phi_v^{t,k} \\ & + \left(D_1\rho^2 - D_2\rho + q\left(G_1\rho\gamma^2 + G_2\rho^3 + G_3\rho^3\right)\right)\sum_{k=0}^{q-1} V_y^{t,k} \\ & + \left(-E_1\rho + q\left(H_1\rho\gamma^2 + H_2\gamma^2\rho + H_3\rho^3\right)\right)\rho\sum_{k=0}^{q-1} V_v^{t,k} \\ & + \left(F_1\gamma^2 - F_2\gamma + q\left(I_1\gamma^3 + I_2\gamma^2\rho + I_3\rho^3\right)\right)\sum_{k=0}^{q-1} V_x^{t,k} \ .\end{aligned}$$

With suitable choice of the step sizes ρ and γ, and constants ψ_y and ψ_v, we get

$$\mathcal{L}^{t,q} - \mathcal{L}^{t,0} \leq -\frac{\gamma}{4}\sum_{k=0}^{q-1} g^{t,k} \ .$$

Then, summing for $t \in \{0, \ldots, T-1\}$ yields the result. □

Note that increasing q allows a faster convergence in terms of iterations but makes each iteration more expensive since the number of oracle calls per iteration is $(2n+3m)+2\times 5(q-1)$. Thus, there is a trade-off between the convergence rate and the overall complexity. In Corollary 5.3.1, we state that the value of q that gives the best sample complexity is $n+m$.

Corollary 5.3.1. *Suppose that Assumptions 5.1 and 5.2 hold. If we take the inner step size* $\rho = \overline{\rho}(n+m)^{-\frac{1}{2}}$*, the outer step size* $\gamma = \min(\overline{\gamma}, \xi\rho)(n+m)^{-\frac{1}{2}}$ *and the inner loop size* $q = n+m$*, then*

$$\mathcal{O}\left((n+m)^{\frac{1}{2}}\varepsilon^{-1} \vee (n+m)\right)$$

calls to oracles are sufficient to find an ε*-stationary point with SRBA.*

Proof. Let us take $\rho = \overline{\rho}(n+m)^{-\frac{1}{2}}$, $\gamma = \min(\xi\rho, \overline{\gamma})$ and $q = n+m$. Then Theorem 5.1 holds:

$$\frac{1}{Tq}\sum_{t=0}^{T-1}\sum_{k=0}^{q-1} g^{t,k} \leq \frac{4}{Tq\gamma}\Gamma^0 \ .$$

with $\Gamma_0 = \mathcal{O}(1)$. To get an ε-stationary solution, we set $T \geq \frac{4}{q\gamma}\Gamma^0\varepsilon^{-1} \vee 1 = \mathcal{O}\left(\frac{1}{q\gamma\varepsilon} \vee 1\right)$. One iteration has $\Theta(q) = \Theta(n+m)$ oracle complexity. As a consequence, the sample complexity to get an ε-stationary point is $\mathcal{O}\left((n+m)^{\frac{1}{2}}\epsilon^{-1} \vee (n+m)\right)$. □

This sample complexity is analogous to the sample complexity of SARAH in the non-convex finite-sum setting. To the best of our knowledge, such a rate is the best known for bilevel empirical risk minimization problems in terms of dependency on the number of samples $n+m$ and the precision ε. This improves by a factor $(n+m)^{-\frac{1}{6}}$ the previous result which was achieved by SABA (Dagréou et al., 2022a). As a comparison, VRBO (Yang et al., 2021) achieves a sample complexity in $\tilde{\mathcal{O}}(\varepsilon^{-\frac{3}{2}})$. Note that, for large value of $n+m$ we can have actually $(n+m)^{\frac{1}{2}}\varepsilon^{-1} \gtrsim \varepsilon^{-2}$. This means that, just like single-level SARAH, the complexity of SRBA can be beaten by others when the number of samples is too high with respect to the desired accuracy (actually if $n+m = \Omega(\varepsilon^{-2})$).

5.4 Lower Bound for Bilevel ERM

In this section, we derive a lower bound for bilevel empirical risk minimization problems. This shows that SRBA is a near-optimal algorithm for this class of problems.

Function and Algorithm Classes. We define the function and algorithm classes we consider.

Definition 5.2. Let n, m two positive integers, $L_{f,1}$ and μ_g two positive constants. The class of the smooth empirical risk minimization problems denoted by $\mathcal{C}^{L_{f,1},\mu_g}$ is the set of pairs of real-valued function families $((f_j)_{1\leq j\leq m}, (g_i)_{1\leq i\leq n})$ defined on $\mathbb{R}^{d_x}\times\mathbb{R}^{d_y}$ such that for all $j\in[m]$, f_j is $L_{f,1}$ smooth and for all $i\in[n]$, g_i is twice differentiable and μ_g-strongly convex.

Note that we consider a class of non-convex bilevel problems. This class contains the functions defining the bilevel formulation of the datacleaning task.

For the algorithmic class, we consider algorithms that use approximate implicit differentiation.

Definition 5.3. Given initial points y^0, v^0, x^0, a *linear bilevel algorithm* $\mathcal{A}$ is a measurable mapping such that for any $((f_j)_{1\leq j\leq m}, (g_i)_{1\leq i\leq n}) \in \mathcal{C}^{L_{f,1},\mu_g}$, the output of $\mathcal{A}((f_j)_{1\leq j\leq m}, (g_i)_{1\leq i\leq n})$ is a sequence $\{(y^t, v^t, x^t, i_t, j_t)\}_{t\geq 0}$ of points (y^t, v^t, x^t) and random variables $i_t \in [n]$ and $j_t \in [m]$ such that for all $t \geq 0$

$$\begin{aligned}
y^{t+1} &\in y^0 + \mathrm{Span}\{\nabla_y g_{i_0}(x^0,y^0), \dots, \nabla_y g_{i_t}(x^t,y^t)\}\\
v^{t+1} &\in v^0 + \mathrm{Span}\{\nabla^2_{yy} g_{i_0}(x^0,y^0)v^0 + \nabla_y f_{j_0}(x^0,y^0),\\
&\qquad \dots, \nabla^2_{yy} g_{i_t}(x^t,y^t)v^t + \nabla_y f_{j_t}(x^t,y^t)\}\\
x^{t+1} &\in x^0 + \mathrm{Span}\{\nabla^2_{xy} g_{i_0}(x^0,y^0)v^0 + \nabla_x f_{j_0}(x^0,y^0),\\
&\qquad \dots, \nabla^2_{xy} g_{i_t}(x^t,y^t)v^t + \nabla_x f_{j_t}(x^t,y^t)\}.
\end{aligned}$$

This algorithm class includes popular stochastic bilevel first-order algorithms, such as AmIGO (Arbel and Mairal, 2022a), FSLA (Li et al., 2022), SOBA, and SABA (Dagréou et al., 2022a). Moreover, despite the projection step, SRBA is part of this algorithm class since the projection of a vector onto Γ is actually just a rescaling.

As a comparison to the existing lower bound for bilevel optimization by Ji and Liang (2023), we consider randomized algorithms and do not assume the value function Φ to be convex or strongly convex.

Main Theorem. Problem (5.1) is actually a smooth non-convex optimization problem. The lower complexity bound for non-convex finite sum problems has been studied by Fang et al. (2018) and Zhou and Gu (2019). In particular, they show that the number of gradient calls needed to get an ε-stationary point for a smooth non-convex finite sum is at least $\mathcal{O}(m^{\frac{1}{2}}\varepsilon^{-1})$, where m is the number of terms in the finite sum.

Intuitively, we expect the lower complexity bound to solve (5.1) to be larger. Indeed, bilevel problems are harder than single-level problems because a bilevel problem involves the resolution of several subproblems to progress in its resolution. Theorem 5.2 formalizes this intuition by showing that the classical single-level lower bound is also a lower bound for bilevel problems.

Theorem 5.2. *For any linear bilevel algorithm $\mathcal{A}$, and any L^f, n, Δ, ε, p such that $\varepsilon \leq (\Delta L^f m^{-1})/10^3$, there exists a dimension $d = \mathcal{O}(\Delta\varepsilon^{-1}m^{\frac{1}{2}}L^f)$, an element $((f_j)_{1\leq j\leq m}, (g_i)_{1\leq i\leq n}) \in \mathcal{C}^{L_f,1,\mu_g}$ such that the value function Φ defined as in (5.1) satisfies $\Phi(x^0) - \inf_{x\in\mathbb{R}^d}\Phi(x) \leq \Delta$ and in order to find $\hat{x} \in \mathbb{R}^d$ such that $\mathbb{E}[\|\nabla\Phi(\hat{x})\|^2] \leq \varepsilon$, $\mathcal{A}$ needs at least $\Omega(m^{\frac{1}{2}}\varepsilon^{-1})$ calls to oracles of the form (5.10).*

The proof is an adaptation of the proof of Zhou and Gu (2019, Theorem 4.7). We take as outer function f defined by $f(x,y) = \sum_{j=1}^m f(U^{(j)}z)$ where f is the "worst-case function" used by Carmon et al. (2021), $U = [U^{(j)}, \dots, U^{(m)}]^\top$ is an orthogonal matrix and $g(x,y) = \frac{1}{2}\|y-x\|^2$. We leverage the fact that $\|\nabla f(y)\|^2 > K$ as long as the two last coordinates of y are zero for some known constant K. Then we use the "zero chain property" to bound the number of indices j such the two last components of $U^{(j)}x^t$ are zero at a given iteration t, implying $\|\nabla\Phi(x^t)\|^2 > \epsilon$ when t is smaller than $\mathcal{O}(m^{\frac{1}{2}}\varepsilon^{-1})$. In the next section, we provide the proof of Theorem 5.2.

5.5 Proof of Theorem 5.2

The proof of Theorem 5.2 is an adaptation of the proof of (Zhou and Gu, 2019, Theorem 4.7) from single-level to bilevel problems. We build the outer function from the worst-case instance of (Zhou and Gu, 2019, Theorem 4.7) and we add a bilevel component by using as inner function the function g defined by $g(x,y) = \frac{\mu_g}{2}\|y-x\|^2$. We start by introducing the different tools used in this proof.

5.5.1 Preliminary results

In what follows, we provide the building blocks of our worst-case instance. The proof uses the following quadratic function presented by (Nesterov, 2018).

Definition 5.4. Let $d \in \mathbb{N}_{>0}$, $\xi \in [0\,,\,+\infty)$ and $\zeta \leq 1$. We define $\mathbf{Q}(.;\xi,d) : \mathbb{R}^d \to \mathbb{R}$ by

$$\mathbf{Q}(x;\xi,d) = \frac{\xi}{2}(x_1-1)^2 + \frac{1}{2}\sum_{k=1}^{d-1}(x_{k+1}-x_k)^2 \ .$$

Proposition 5.5 proposition comes directly from (Zhou and Gu, 2019, Proposition 3.5). The first part of the proposition gives us the regularity of $\mathbf{Q}$. The second part shows that a function defined as $\mathbf{Q}(U\times\cdot;\xi,d) + \sum_{p=1}^q g(\langle u_p,\cdot\rangle)$ verifies the so-called "zero-chain property" (Carmon et al., 2020): if $Ux \in \mathrm{Span}(u_1,\dots,u_k)$, then we gain a non-zero coordinate by calling the gradient

$$\nabla\left[\mathbf{Q}\left(U\times\cdot;\xi,d\right) + \sum_{p=1}^q g(\langle u_p,\cdot\rangle)\right](x) \ .$$

In other words, that makes us progress in the problem resolution.

Proposition 5.5. *For $d \in \mathbb{N}_{>0}$, $\xi \in [0\,,\,+\infty)$ and $\zeta \leq 1$. The following holds:*

1. *$\mathbf{Q}(\,\cdot\,;\xi,d)$ is convex and 4-smooth.*
2. *Let $q \in \mathbb{N}_{>0}$, $U = [u_1,\dots,u_d]^\top \in \mathbb{R}^{d\times q}$ such that $UU^\top = I$, and for $k \leq d$, let*

$$U^{(\leq k)} \;=\; [u_1,\dots,u_k,0,\dots,0]^\top \in \mathbb{R}^{d\times q} \;\;.$$

Let $g : \mathbb{R} \to \mathbb{R}$ differentiable such that $g'(0) = 0$. Then for any $x \in \mathbb{R}^q$ such that $Ux = U^{(\leq k)}x$, then

$$\nabla\left[\mathbf{Q}(U\times\,\cdot\,;\xi,d) + \sum_{p=1}^{d} g(\langle u_p,\,\cdot\,\rangle)\right](x) \in \mathrm{Span}(u_1,\dots,u_k,u_{k+1}) \;\;.$$

Proof. Let $x \in \mathbb{R}^q$ such that $Ux = U^{(\leq k)}x$. For $0 \leq k \leq d$, we denote

$$\mathbb{R}^{k,d} = \{v \in \mathbb{R}^d, v_{k+1} = \cdots = v_d = 0\} \;\;.$$

Let us write $\mathbf{Q}(x;\xi,d) = \frac{1}{2}x^\top Ax + b^\top x + c$ with

$$A = \begin{bmatrix} 1+\xi & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix} \in \mathbb{R}^{d\times d} \;\;,$$

$b = \xi(1,0,\dots,0)^\top$ and $c = \frac{\xi}{2}(1,0,\dots,0)^\top$.

On the one hand it is known from (Nesterov, 2018, Lemma 2.5.1) that if $v \in \mathbb{R}^{k,d}$,

$$\nabla\mathbf{Q}(v;\xi,d) \in \mathbb{R}^{k+1,d}$$

As a consequence,

$$\nabla\mathbf{Q}(Ux;\xi,d) = \nabla\mathbf{Q}(\underbrace{U^{(\leq k)}}_{\in\mathbb{R}^{k,d}}x;\xi,d) \in \mathbb{R}^{k+1,d}$$

and

$$\nabla[\mathbf{Q}(U\times\,\cdot\,;\xi,d)](x) = U^\top\nabla\mathbf{Q}(Ux;\xi,d) \in \mathrm{Span}(u_1,\dots,u_{k+1}) \;\;.$$

On the other hand,

$$\nabla\left[\sum_{p=1}^{d} g(\langle u_p,\,\cdot\,\rangle)\right](x) = \sum_{p=1}^{d} g'(\langle u_p,x\rangle)u_p = \sum_{p=1}^{k} g'(\langle u_p,x\rangle)u_p \in \mathrm{Span}(u_1,\dots,u_{k+1}) \;\;.$$

Thus

$$\nabla\left[\mathbf{Q}(U\times\,\cdot\,;\xi,d) + \sum_{p=1}^{d} g(\langle u_p,\,\cdot\,\rangle)\right](x) \in \mathrm{Span}(u_1,\dots,u_k,u_{k+1}) \;\;.$$

□

However, the function $\mathbf{Q}$ is convex. That is why we also use the function Γ introduced by Carmon et al. (2021). As explained by Carmon et al. (2021), this function is essential to lower bound the gradient of our worst wase instance.

Definition 5.5. Let $d \in \mathbb{N}_{>0}$. We define $\Gamma(\,\cdot\,;d) : \mathbb{R}^{d+1} \to \mathbb{R}$ by

$$\Gamma(x;d) = 120\sum_{k=1}^{d}\int_1^{x_k} \frac{t^2(t-1)}{1+t^2}\mathrm{d}t \;\;.$$

An important property of Γ shown in (Carmon et al., 2021, Lemma 2) is the smoothness of the function Γ.

Proposition 5.6. *There exists a constant $c > 0$ such that $\Gamma(\,\cdot\,; d)$ is c-smooth.*

Now we introduce the function f_{nc} we use to build our worst-case instance. This function comes from (Zhou and Gu, 2019, Definition 3.5). It is the sum of the quadratic function defined in Definition 5.4 and the non-convex component given in Definition 5.5.

Definition 5.6. For $\alpha > 0$ and $d \in \mathbb{N}_{>0}$, $f_{\mathrm{nc}}(\,\cdot\,; \alpha, d) : \mathbb{R}^{d+1} \to \mathbb{R}$ is defined a

$$f_{\mathrm{nc}}(x; \alpha, d) = \mathbf{Q}(x; \sqrt{\alpha}, d+1) + \alpha\Gamma(x) \ .$$

The essential properties of f_{nc} come from (Carmon et al., 2021, Lemmas 2, 3, 4). The first part provides the regularity properties of f_{nc}. The second part bounds the distance between $f_{\mathrm{nc}}(\,\cdot\,; \alpha, d)$ and the optimal value of the function. The third part will be key to the overall proof. In words, it states that as long $x \in \mathbb{R}^{d+1}$ has its two last components equal to zero, the norm of the gradient of $f_{\mathrm{nc}}(\,\cdot\,; \alpha, d)$ is higher than a constant controlled by α. As a consequence, if α is properly chosen, as soon as $x_d = x_{d+1} = 0$, we are ensured that $\|\nabla f_{\mathrm{nc}}(x; \alpha, d)\| \geq \epsilon$.

Proposition 5.7. *For $\alpha \in [0\,,\,1]$, it holds*

1. $-\alpha c \preceq \nabla^2 f_{\mathrm{nc}} \preceq 4 + \alpha c$.
2. $f_{\mathrm{nc}}(0; \alpha, d) - \inf_x f_{\mathrm{nc}}(x; \alpha, d) \leq \frac{\sqrt{\alpha}}{2} + 10\alpha d$.
3. *For $x \in \mathbb{R}^{d+1}$ such that $x_d = x_{d+1} = 0$, $\|\nabla f_{\mathrm{nc}}(x; \alpha, d)\| \geq \frac{\alpha^{\frac{3}{4}}}{4}$.*

From now we denote

$$\mathcal{O}(a, b) = \{U \in \mathbb{R}^{a \times b}, UU^\top = I_a\} \ .$$

The following Lemma adapted from (Zhou and Gu, 2019) is fundamental for our lower bound proof.

Lemma 5.4. *Let $d, m \in \mathbb{N}_{>0}$ and $U \in \mathcal{O}((d+1)m, (d+1)m)$. We denote*

$$U = \begin{bmatrix} U^{(1)} \\ \vdots \\ U^{(m)} \end{bmatrix}$$

with each $U^{(i)} \in \mathcal{O}(d+1,\ (d+1)m)$. Let $\{\Phi_j\}_{j \in [m]}$ with $\Phi_j(x) = f_{\mathrm{nc}}(U^{(j)}x; \alpha, d)$ and $\Phi = \frac{1}{m}\sum_{j=1}^m \Phi_j$. Let $x \in \mathbb{R}^{(d+1)m}$ and $z^{(j)} = U^{(j)}x \in \mathbb{R}^{d+1}$. Let $\mathcal{I} = \{j \in [m], z_d^{(j)} = z_{d+1}^{(j)} = 0\}$. Then it holds

$$\|\nabla\Phi(x)\|^2 \geq \frac{\alpha^{\frac{3}{2}}|\mathcal{I}|}{16m^2} \ .$$

Proof. We have

$$\begin{aligned}\|\nabla\Phi(x)\|^2 &= \left\|\frac{1}{m}\sum_{j=1}^{d}\nabla\Phi_j(x)\right\|^2 \\ &= \left\|\frac{1}{m}\sum_{j=1}^{m}(U^{(j)})^\top\nabla f_{\mathrm{nc}}(U^{(j)}x;\alpha,d)\right\|^2 \\ &= \frac{1}{m^2}\sum_{j=1}^{m}\left\|(U^{(j)})^\top\nabla f_{\mathrm{nc}}(U^{(j)}x;\alpha,d)\right\|^2 \\ &\quad + \frac{2}{m^2}\sum_{\substack{j,l=1\\ j\neq l}}^{m}\nabla f_{\mathrm{nc}}(U^{(j)}x;\alpha,d)^\top U^{(l)}(U^{(j)})^\top\nabla f_{\mathrm{nc}}(U^{(j)}x;\alpha,d) \\ &= \frac{1}{m^2}\sum_{j=1}^{m}\left\|(U^{(j)})^\top\nabla f_{\mathrm{nc}}(U^{(j)}x;\alpha,d)\right\|^2\end{aligned}$$

where the last equality comes from the fact that we have for for any $j \neq l$, $U^{(l)}(U^{(j)})^\top = 0$ since $U \in \mathcal{O}\left((d+1)m,(d+1)m\right)$. Now, using the third part of Proposition 5.7, we get

$$\begin{aligned}\|\nabla\Phi(x)\|^2 &\geq \frac{1}{m^2}\sum_{j\in\mathcal{I}}\left\|(U^{(j)})^\top\nabla f_{\mathrm{nc}}(U^{(j)}x;\alpha,d)\right\|^2 \\ &\geq \frac{1}{m^2}\sum_{j\in\mathcal{I}}\left\|\nabla f_{\mathrm{nc}}(U^{(j)}x;\alpha,d)\right\|^2 \\ &\geq \frac{\alpha^{\frac{3}{2}}|\mathcal{I}|}{16m^2}\ .\end{aligned}$$

□

5.5.2 Main proof

Now we are ready to prove Theorem 5.2.

Proof. We consider $U \in \mathcal{O}((T+1)m,(T+1)m)$ and we denote

$$U = \begin{bmatrix} U^{(1)} \\ \vdots \\ U^{(m)} \end{bmatrix}$$

with $U^{(j)} = (u_1^{(j)},\dots,u_{T+1}^{(j)})^\top \in \mathcal{O}(T+1,(T+1)m)$.

For $j\in[m]$, we choose $\overline{f}_j : \mathbb{R}^{(T+1)m+(T+1)m} \to \mathbb{R}$ defined by

$$\overline{f}_j(x,y) = f_{\mathrm{nc}}(U^{(j)}y;\alpha,T)$$

and we set $\overline{f} = \frac{1}{m}\sum_{j=1}^{m}\overline{f}_j$. We also define for $i\in[n]$ $\overline{g}_i(x,y) = \frac{1}{2}\|y-x\|^2$, $\overline{g} = \frac{1}{n}\sum_{i=1}^{n}\overline{g}_i$, $\overline{y}^*(x) = \arg\min_y \overline{g}(x,y)$ and $\overline{\Phi}(x) = \overline{f}(x,\overline{y}^*(x)) = f_{\mathrm{nc}}(U^{(j)}x;\alpha,T)$. By Proposition 5.7, $\overline{f}_j$ is $4+\frac{\alpha c}{m}$ smooth, and $\overline{g}_i$ is 1-smooth and 1-strongly convex.

We have

$$\overline{\Phi}(0) - \inf_x \overline{\Phi}(x) \leq \sqrt{\alpha} + 10\alpha T\ .$$

We finally consider $f_j(x,y) = \lambda_f\overline{f}_j(x/\beta,y/\beta)$, $g_i(x,y) = \lambda_G\overline{g}_i(x/\beta,y/\beta)$. As a consequence, we have $y^*(x) = \arg\min_y g(x,y) = \overline{y}^*(x)$ and $\Phi(x) = f(x,y^*(x))$. We also consider a *fixed* indices sequence (i_t,j_t). We set

$$\alpha = \min\left\{1,\frac{m}{c}\right\},\quad \lambda_f = \frac{160m\epsilon}{L_{f,1}\alpha^{3/2}},\quad \beta = \sqrt{5\lambda_f/L_{f,1}},\quad \lambda_G = \beta^2\mu_g,\quad T = \frac{\Delta L_{f,1}}{1760m\epsilon}\sqrt{\alpha}\ .$$

We can check that each f_j is $L_{f,1}$-smooth, and each g_i is μ_g-strongly convex.

Assuming $\epsilon \leq \Delta L_{f,1}\alpha/(1760m)$ ensures that $\Phi(0) - \inf_x \Phi(x) \leq \Delta$ (we can check that $\Phi(0) = \lambda_f \overline{\Phi}(0)$ and $\inf \Phi = \lambda_f \inf \overline{\Phi}$).

Let us assume without loss of generality that the algorithm at initialization we have $y^0 = v^0 = x^0 = 0$ and consider (y^t, v^t, x^t) the output of an algorithm with the known sequence (i_t, j_t).

Given our inner function and the fact that $\nabla_x f(x,y) = 0$ for any $(x,y) \in \mathbb{R}^{(m+1)d+(m+1)d}$, we have

$$y^{t+1} \in \mathrm{Span}(y^0 - x^0, \dots, y^t - x^t) \tag{5.22}$$
$$v^{t+1} \in \mathrm{Span}(v^0 + \nabla_y f_{j_0}(x^0, y^0), \dots, v^t + \nabla_y f_{j_t}(x^t, y^t)) \tag{5.23}$$
$$x^{t+1} \in \mathrm{Span}(v^0, \dots, v^t) \ . \tag{5.24}$$

Since $v^0 = 0$, we have by Equation (5.23) $v^1 \in \mathrm{Span}(\nabla_y f_{j_0}(x^0, y^0))$ and by induction

$$v^{t+1} \in \mathrm{Span}(\nabla_y f_{j_0}(x^0, y^0), \dots, \nabla_y f_{j_t}(x^t, y^t)) \ .$$

Therefore, using Equation (5.24), we have

$$x^{t+1} \in \mathrm{Span}(\nabla_y f_{j_0}(x^0, y^0), \dots, \nabla_y f_{j_t}(x^t, y^t)) \ .$$

Since $y^0 = 0$, by Equation (5.22), $y^1 \in \mathrm{Span}(x^0)$ and by induction

$$y^{t+1} \in \mathrm{Span}(x^0, \dots, x^t) \ .$$

As a consequence,

$$x^t \in \mathrm{Span}(\nabla_y f_{j_0}(x^0, \mathrm{Span}(x^0)), \dots, \nabla_y f_{j_t}(x^t, \mathrm{Span}((x^s)_{s \leq t}))) \ .$$

Let us denote $z^{(j,t)} = U^{(j)} x^t$. Since $x^0 = 0$, $z^{(j_0,0)} = 0$ and by the second part of Proposition 5.5, $x^1 \in \mathrm{Span}(u_1^{(j_0)})$.

Now we assume that for all $s \leq t$ we have

$$x^s \in \mathrm{Span}(u_1^{(j_0)}, \dots, u_s^{(j_0)}, \dots, u_1^{(j_{s-1})}, \dots, u_s^{(j_{s-1})}) \ .$$

There exist scalars $\alpha_1, \dots, \alpha_r, \beta_{1,1}, \beta_{2,1}, \beta_{2,2}, \dots, \beta_{t,1}, \dots, \beta_{t,t}$ such that

$$x^{t+1} = \sum_{r=1}^{t} \alpha_r \nabla_y f_{j_r} \left(x^r, \sum_{s=1}^{r} \beta_{r,s} x^s \right) \ .$$

Let $X^r = \sum_{s=1}^{r} \beta_{r,s} x^s$. For $r \in \{1, \dots, t\}$, we have by induction hypothesis

$$X^r \in \mathrm{Span}(u_1^{(j_0)}, \dots, u_r^{(j_0)}, \dots, u_1^{(j_{r-1})}, \dots, u_r^{(j_{r-1})}) \ .$$

By orthogonality, we have

$$\mathrm{Span}(u_1^{(j_0)}, \dots, u_r^{(j_0)}, \dots, u_1^{(j_{r-1})}, \dots, u_r^{(j_{r-1})}) \perp \mathrm{Span}(u_{r+1}^{(j_r)}, \dots, u_{T+1}^{(j_r)}) \ .$$

As a consequence

$$U^{(j_r)} X^r = (\langle u_1^{(j_r)}, X^r \rangle, \dots, \langle u_r^{(j_r)}, X^r \rangle, 0, \dots, 0) \ .$$

We can use Proposition 5.5 to say

$$\nabla_y f_{j_r}(x^r, X^r) \in \mathrm{Span}(u_1^{(j_r)}, \dots, u_{r+1}^{(j_r)}) \subset \mathrm{Span}(u_1^{(j_0)}, \dots, u_r^{(j_0)}, u_{r+1}^{(j_0)}, \dots, u_1^{(j_r)}, \dots, u_{r+1}^{(j_r)}) \ .$$

And we get finally

$$x^{t+1} \in \mathrm{Span}(u_1^{(j_0)}, \dots, u_t^{(j_0)}, u_{t+1}^{(j_0)}, \dots, u_1^{(j_t)}, \dots, u_{t+1}^{(j_t)}) \ .$$

By induction, for any t, we have

$$x^t \in \mathrm{Span}(\underbrace{u_1^{(j_0)}, \dots, u_t^{(j_0)}, \dots, u_1^{(j_t)}, \dots, u_t^{(j_t)}}_{\text{at most } mt \text{ vectors}})$$

and so

$$x^t \perp \mathrm{Span}((u_1^{(j)}, \dots, u_{T+1}^{(j)})_{j\in[m]\setminus\{j_0,\dots,j_t\}}, (u_{t+1}^{(j)}, \dots, u_{T+1}^{(j)})_{j\in\{j_0,\dots,j_t\}}) \ .$$

As a consequence, for $t \leq \frac{m}{2}T$, let $\mathcal{I} = \{j, z_T^{(j,t)} = z_{T+1}^{(j,t)} = 0\}$ with $z^{(j,t)} = U^{(j)}x^t$. Since $t \leq \frac{m}{2}T$, we have $|\mathcal{I}| \leq \frac{m}{2}$ and by Lemma 5.4, we have

$$\|\nabla\Phi(x^t)\| \geq \epsilon \ .$$

If we define $T((x^t)_t, \Phi) = \inf\{t \in \mathbb{N}, \|\nabla\Phi(x^t)\|^2 \leq \epsilon\}$, we just showed that for the fixed sequence (i_t, j_t), we have

$$T((x^t)_t, \Phi) \geq \frac{m}{2}T = \Omega(\sqrt{m}\epsilon^{-1}) \ .$$

The right-hand side being independent from the sequence (i_t, j_t), for $t \leq \frac{m}{2}T$, we have

$$\mathbb{E}[\|\nabla\Phi(x^t)\|^2] > \epsilon$$

where the expectation is taken over the random choice of $i_0, \dots, i_{t-1}, j_0, \dots, j_{t-1}$. □

5.6 Numerical Experiments

Even though our contribution is mostly theoretical, we run several experiments to highlight to compare the proposed algorithm with state-of-the-art stochastic bilevel solvers. We compare our method to AmIGO (Arbel and Mairal, 2022a), F^{2}SA (Kwon et al., 2023a), MRBO (Yang et al., 2021), VRBO (Yang et al., 2021), StocBiO (Ji et al., 2021) and SABA (Dagréou et al., 2022a). They are run on a synthetic problem with quadratic functions and on a hyperparameter selection problem for ℓ^2-regularized logistic regression with the dataset IJCNN1[1]. A more detailed description of the experiments is available in Section B.2.

Experiments on quadratics. To evaluate the performance of stochastic bilevel optimizers in a controlled setting, we perform a benchmark on quadratic loss functions described in Section B.2. Here f and g are quadratic jointly in (y, x), allowing us to choose freely the conditioning of f, g, and Φ. We take for the Hessian and cross derivative matrices of each sample, the empirical correlation of random vectors drawn with a prescribed covariance matrix. The generation process is detailed in Section B.2. In figure 5.1, we report the norm of the gradient of the value function function with respect to time. Our first observation is that among all the methods, SRBA and SABA converge the fastest. These two solvers share two key ingredients: variance reduction and warm-starting. Variance reduction makes the variance of the gradient estimate go to zero without using decreasing step sizes. The warm-starting strategy in both the approximation of $y^*(x^t)$ and the approximation of $v^*(x^t)$ enables getting an estimator of $\nabla\Phi(x^t)$ which is asymptotically unbiased, without requiring an increasing number of inner iterations or batch-size. Note that solvers using Neumann iterations (VRBO, MRBO, stocBiO) fail to converge because Neumann iterations provide a biased estimate of $v^*(x)$. Moreover, AmIGO and stocBiO evolve slowly after some iterations because they require vanishing step sizes to converge. Finally, SRBA is faster than SABA, which is consistent with the theory.

Hyperparameter selection. We also run an experiment on the hyperparameter selection problem for ℓ^2-regularized logistic regression with the IJCNN1 dataset. SRBA shows good performances in the experiment, both in speed and accuracy. It is competitive with other state-of-the-art methods AmIGO and SABA, while going faster than Amigo and requiring less memory than SABA. VRBO –another extension of SARAH for bilevel problems– is slower in all problems. This is due to the burden of computing the approximate hypergradient at each inner iteration without updating the outer

[1] \Phittps://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

parameter. We can also notice that in the experiment on IJCNN1, the slowest method are method implementing Neumann approximations to approximate $v^*(x)$. Note that this last experiment does not include F^{2}SA because we find that on this problem, the norm of the iterates of F^{2}SA goes towards infinity.

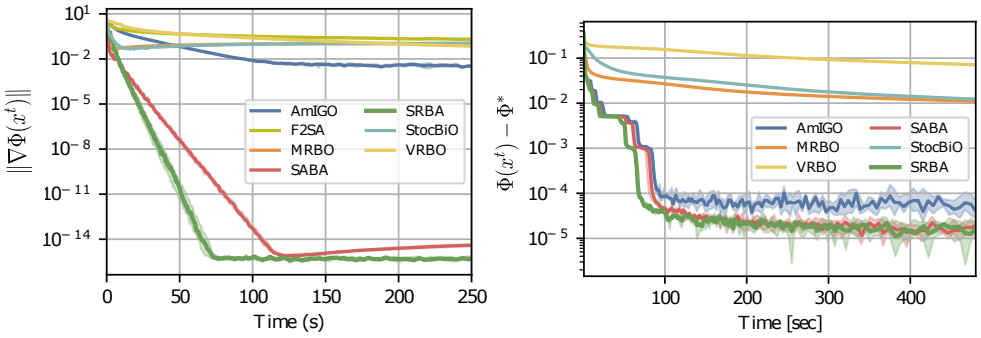


Figure 5.1: Comparison of the behavior of SRBA with other stochastic bilevel solvers. For each experiment, the solvers are run with 10 different seeds, and the median performance over these seeds is reported. The shaded area corresponds to the performances between the 20% and the 80% percentiles. The performances are reported with respect to wall clock time. **Left:** Experiments on quadratic functions. We report the gradient norm of the value function. **Right:** Hyperparameter selection with the IJCNN1 dataset.

5.7 Conclusion

In this chapter, we have introduced SRBA, an algorithm for bilevel empirical risk minimization. We have demonstrated that the sample complexity of SRBA is $\mathcal{O}((n+m)^{\frac{1}{2}}\varepsilon^{-1})$ for any bilevel problem where the inner problem is strongly convex. Then, we have demonstrated that any bilevel empirical risk minimization algorithm has a sample complexity of at least $\mathcal{O}(m^{\frac{1}{2}}\varepsilon^{-1})$ on some problems where the inner problem is strongly convex. This demonstrates that SRBA is optimal, up to constant factors, and that bilevel ERM is as hard as single-level non-convex ERM.

CHAPTER B

APPENDIX TO LOWER BOUND FOR BILEVEL EMPIRICAL RISK MINIMIZATION

Section B.1 contains the necessary lemmas and proofs of Section 5.3. Section 5.5 contains the proof of the lower bound for stochastic bilevel optimization. Section B.2 details the setting of the numerical experiments.

B.1 Convergence analysis of SRBA

B.1.1 Technical lemmas

Lemma B.1. *There exists constant L_{y^*} and L_{v^*} such that for any $x_1, x_2 \in \mathbb{R}^d$, we have*

$$\|y^*(x_1) - y^*(x_2)\| \leq L_{y^*}\|x_1 - x_2\| \quad \textit{and} \quad \|v^*(x_1) - v^*(x_2)\| \leq L_{v^*}\|x_1 - x_2\|$$

Proof. The Jacobian of y^* reads $\mathrm{d}y^*(x) = [\nabla^2_{yy} g(x, y^*(x))]^{-1}\nabla^2_{yx} g(x, y^*(x))$. By μ_g-strong convexity and $L_{g,1}$-smoothness of g, we have $\|\mathrm{d}y^*(x)\| \leq \frac{L_{g,1}}{\mu_g}$ which implies that y^* is L_{y^*}-Lipschtiz with $L_{y^*} = \frac{L_{g,1}}{\mu_g}$.

For v^* we do the computation directly:

$$\begin{aligned}
\|v^*(x_1) - v^*(x_2)\| &= \|[\nabla^2_{yy} g(y^*(x_1), x_1)]^{-1}\nabla_y f(y^*(x_1), x_1) \\
&\quad - [\nabla^2_{yy} g(y^*(x_2), x_2)]^{-1}\nabla_y f(y^*(x_2), x_2)\| \\
&\leq \|[\nabla^2_{yy} g(y^*(x_1), x_1)]^{-1}(\nabla_y f(y^*(x_1), x_1) - \nabla_y f(y^*(x_2), x_2))\| \\
&\quad + \|([\nabla^2_{yy} g(y^*(x_1), x_1) - [\nabla^2_{yy} g(y^*(x_2), x_2)]^{-1}]^{-1}\nabla_y f(y^*(x_2), x_2)\| \\
&\leq \left(\frac{L_{f,1}}{\mu_g} + \frac{L_{g,2}L_{f,0}}{\mu_g^2}\right)\|(y^*(x_1), x_1) - (y^*(x_2), x_2)\| \\
&\leq \left(\frac{L_{f,1}}{\mu_g} + \frac{L_{g,2}L_{f,0}}{\mu_g^2}\right)(\|y^*(x_1) - y^*(x_2)\| + \|x_1 - x_2\|) \\
&\leq \left(1 + \frac{L_{g,1}}{\mu_g}\right)\left(\frac{L_{f,1}}{\mu_g} + \frac{L_{g,2}L_{f,0}}{\mu_g^2}\right)\|x_1 - x_2\|
\end{aligned}$$

Then taking $L_{v^*} = \left(1 + \frac{L_{g,1}}{\mu_g}\right)\left(\frac{L_{f,1}}{\mu_g} + \frac{L_{g,2}L_{f,0}}{\mu_g^2}\right)$ concludes the proof. □

Lemma B.2. *Let us consider the update directions $D_y^{t,k} = \boldsymbol{\Delta}_y^{t,k}/\rho$, $D_v^{t,k} = \boldsymbol{\Delta}_v^{t,k}/\rho$ and $D_x^{t,k} = \boldsymbol{\Delta}_x^{t,k}/\gamma$*

where $\boldsymbol{\Delta}_y^{t,k}$, $\boldsymbol{\Delta}_v^{t,k}$ *and* $\boldsymbol{\Delta}_x^{t,k}$ *verify Equations* (5.6) *to* (5.8). *Then it holds*

$$\mathbb{E}[\|D_y^{t,k} - D_y(\mathbf{u}^{t,k})\|^2] \leq \sum_{r=1}^{k} L_{g,1}(\rho^2 \mathbb{E}[\|D_y^{t,r-1}\|^2] + \gamma^2 \mathbb{E}[\|D_y^{t,r-1}\|^2])$$

$$\begin{aligned}\mathbb{E}[\|D_v^{t,k} - D_v(\mathbf{u}^{t,k})\|^2] \leq\ & 4\rho^2 \left((L_{g,2}R)^2 + (L_{f,1})^2\right) \sum_{r=1}^{k} \mathbb{E}[\|D_y^{t,r-1}\|^2] + 4\rho^2 (L_{g,1})^2 \sum_{r=1}^{k} \mathbb{E}[\|\mathcal{G}_v^{t,r-1}\|^2] \\ & + 4\gamma^2 \left((L_{g,2}R)^2 + (L_{f,1})^2\right) \sum_{r=1}^{k} \mathbb{E}[\|D_x^{t,r-1}\|^2]\end{aligned}$$

$$\begin{aligned}\mathbb{E}[\|D_x^{t,k} - D_x(\mathbf{u}^{t,k})\|^2] \leq\ & 4\rho^2 \left((L_{g,2}R)^2 + (L_{f,1})^2\right) \sum_{r=1}^{k} \mathbb{E}[\|D_y^{t,r-1}\|^2] + 4\rho^2 (L_{g,1})^2 \sum_{r=1}^{k} \mathbb{E}[\|\mathcal{G}_v^{t,r-1}\|^2] \\ & + 4\gamma^2 \left((L_{g,2}R)^2 + (L_{f,1})^2\right) \sum_{r=1}^{k} \mathbb{E}[\|D_x^{t,r-1}\|^2] \ .\end{aligned}$$

Proof. **Direction** D_y

We start from Proposition 5.4.

$$\begin{aligned}\mathbb{E}[\|D_y^{t,k} - D_y(\mathbf{u}^{t,k})\|^2] &= \mathbb{E}[\|D_y^{t,k} - \nabla_y g(y^{t,k}, x^{t,k})\|^2] \\ &= \sum_{r=1}^{k} \mathbb{E}[\|D_y^{t,r} - D_y^{t,r-1}\|^2] - \sum_{r=1}^{k} \mathbb{E}[\|\nabla_y g(y^{t,r}, x^{t,r}) - \nabla_y g(y^{t,r-1}, x^{t,r-1})\|^2] \\ &\leq \sum_{r=1}^{k} \mathbb{E}[\|D_y^{t,r} - D_y^{t,r-1}\|^2] \\ &\leq \sum_{r=1}^{k} L_{g,1}(\rho^2 \mathbb{E}[\|D_y^{t,r-1}\|^2] + \gamma^2 \mathbb{E}[\|D_y^{t,r-1}\|^2])\end{aligned}$$

where the last inequality comes from the smoothness of each g_i.

Direction D_v For D_v, the proof is almost the same. Proposition 5.4 gives us

$$\mathbb{E}[\|D_v^{t,k} - D_v(\mathbf{u}^{t,k})\|^2] \leq \sum_{r=1}^{k} \mathbb{E}[\|D_v^{t,r} - D_v^{t,r-1}\|^2] \ .$$

Then, using the boundedness of v and regularity of each g_i and F_j, we have

$$\begin{aligned}\mathbb{E}[\|D_v^{t,r} - D_v^{t,r-1}\|^2] &\leq 2(\mathbb{E}[\|\nabla_{yy}^2 g_i(y^{t,r}, x^{t,r}) v^{t,r} - \nabla_{yy}^2 g_i(y^{t,r-1}, x^{t,r-1}) v^{t,r-1}\|^2] \\ &\quad + \mathbb{E}[\|\nabla_x F_j(y^{t,r}, x^{t,r}) - \nabla_x F_j(y^{t,r-1}, x^{t,r-1})\|^2]) \\ &\leq 4(\mathbb{E}[\|\nabla_{yy}^2 g_i(y^{t,r}, x^{t,r})(v^{t,r} - v^{t,r-1})\|^2] \\ &\quad + \mathbb{E}[\|(\nabla_{yy}^2 g_i(y^{t,r}, x^{t,r}) - \nabla_{yy}^2 g_i(y^{t,r-1}, x^{t,r-1})) v^{t,r-1}\|^2] \\ &\quad + (L_{f,1})^2(\gamma^2 \mathbb{E}[\|D_y^{t,r-1}\|] + \rho^2 \mathbb{E}[\|D_x^{t,r-1}\|^2])) \\ &\leq 4((L_{g,1})^2 \rho^2 \mathbb{E}[\|\mathcal{G}_v^{t,r-1}\|^2] \\ &\quad + (L_{g,2})^2 R^2(\rho^2 \mathbb{E}[\|D_y^{t,r-1}\|] + \gamma^2 \mathbb{E}[\|D_x^{t,r-1}\|^2]) \\ &\quad + (L_{f,1})^2(\rho^2 \mathbb{E}[\|D_y^{t,r-1}\|] + \gamma^2 \mathbb{E}[\|D_x^{t,r-1}\|^2])) \\ &\leq 4\rho^2 \left((L_{g,2}R)^2 + (L_{f,1})^2\right) \mathbb{E}[\|D_y^{t,r-1}\|^2] + 4\rho^2 (L_{g,1})^2 \mathbb{E}[\|\mathcal{G}_v^{t,r-1}\|^2] \\ &\quad + 4\gamma^2 \left((L_{g,2}R)^2 + (L_{f,1})^2\right) \mathbb{E}[\|D_x^{t,r-1}\|^2] \ .\end{aligned}$$

Direction D_x The proof is the same as the proof for D_v. □

B.1.2 A technical lemma for Lemma 5.1

Let $\phi_y(y,x) = g(y,x) - g(x,y^*(x))$ the inner suboptimality gap. The proof of Lemma 5.1 is based on the smoothness of ϕ_y, which is the object of the following lemma.

Lemma B.3. *The function ϕ_y has Λ_y-Lipschitz continuous gradient on $\mathbb{R}^p \times \mathbb{R}^d$, for some constant Λ_y.*

Proof. For any $(y,x) \in \mathbb{R}^p \times \mathbb{R}^d$, we have

$$\nabla_y \phi_y(y,x) = \nabla_y g(y,x) \text{ and } \nabla_x \phi_y(y,x) = \nabla_x g(y,x) - \nabla_x g(x,y^*(x)) \ .$$

Let us consider $(y,x) \in \mathbb{R}^p \times \mathbb{R}^d$ and $(y',x') \in \mathbb{R}^p \times \mathbb{R}^d$. Since ∇g is $L_{g,1}$-Lipschitz continuous, we have directly

$$\|\nabla_y \phi_y(y,x) - \nabla_y \phi_y(y',x')\| \leq L_{g,1} \|(y,x) - (y',x')\| \ .$$

Moreover, we have

$$\begin{aligned}
\|\nabla_x \phi_y(y,x) - \nabla_x \phi_y(y',x')\| &\leq \|\nabla_x g(y,x) - \nabla_x g(y',x')\| \\
&\quad + \|\nabla_x g(x,y^*(x)) - \nabla_x g(y^*(x'),x')\| \\
&\leq L_{g,1}\|(y,x) - (y',x')\| + L_{g,1}\|(x,y^*(x)) - (y^*(x'),x')\| \\
&\leq L_{g,1}\|(y,x) - (y',x')\| + L_{g,1}(\|y^*(x) - y^*(x')\| + \|x - x'\|) \ .
\end{aligned}$$

From Lemma B.1, y^* is L_* Lipschitz continuous, so

$$\begin{aligned}
\|\nabla_x \phi_y(y,x) - \nabla_x \phi_y(y',x')\| &\leq L_{g,1}\|(y,x) - (y',x')\| + L_{g,1}(\|y^*(x) - y^*(x')\| + \|x - x'\|) \\
&\leq L_{g,1}\|(y,x) - (y',x')\| + L_{g,1}(L_* + 1)\|x - x'\| \\
&\leq L_{g,1}(L_{y^*} + 2)\|(y,x) - (y',x')\| \ .
\end{aligned}$$

As a consequence

$$\begin{aligned}
\|\nabla \phi_y(y,x) - \nabla \phi_y(y',x')\| &\leq \|\nabla_y \phi_y(y,x) - \nabla_y \phi_y(y',x')\| + \|\nabla_x \phi_y(y,x) - \nabla_x \phi_y(y',x')\| \\
&\leq L_{g,1}(L_{y^*} + 3)\|(y,x) - (y',x')\| \ .
\end{aligned}$$

Hence, ϕ_y is Λ_y smooth with $\Lambda_y = L_{g,1}(L_{y^*} + 3)$. □

B.1.3 Proof of Lemma 5.2

Recall that we denote

$$\Psi(y,v,x) = \frac{1}{2} v^\top \nabla^2_{yy} g(y,x) v + \nabla_y f(y,x)^\top v$$

and

$$\phi_v(v,x) = \Psi(y^*(x),v,x) - \Psi(y^*(x),v^*(x),x) \ .$$

As for Lemma 5.1, the key property we need is the smoothness of ϕ_v. The derivatives of ϕ_v involve the third derivative of g. For a tensor $T \in \mathbb{R}^{p_1 \times p_2 \times p_3}$ and a vector $a \in \mathbb{R}^{p_3}$ we denote $(T|a)$ the matrix in $\mathbb{R}^{p_1 \times p_2}$ defined by:

$$(T|a) = \left[\sum_{k=1}^{p_3} T_{i,j,k} a_k \right]_{\substack{1 \leq i \leq p_1 \\ 1 \leq j \leq p_2}} \ .$$

Lemma B.4. *The function ϕ_v has Λ_v-Lipschitz continuous gradient on $\Gamma \times \mathbb{R}^d$, for some constant Λ_v.*

Proof. For any $(v,x) \in \Gamma \times \mathbb{R}^d$, we have

$$\nabla_y \phi_v(v,x) = D_v(y^*(x),v,x)$$

and

$$\begin{aligned}\nabla_x \phi_v(v,x) = (\mathrm{d}y^*(x))^\top \Big[&\frac{1}{2}(\nabla^3_{111} g(x,y^*(x))|v)v - \frac{1}{2}(\nabla^3_{111} g(x,y^*(x))|v^*(x))v^*(x) \\ &+\nabla^2_{yy} f(x,y^*(x))v - \nabla^2_{yy} f(x,y^*(x))v^*(x)\Big] \\ &+ \Big[\frac{1}{2}(\nabla^3_{211} g(x,y^*(x))|v)v - \frac{1}{2}(\nabla^3_{211} g(x,y^*(x))|v^*(x))v^*(x) \\ &+\nabla^2_{21} f(x,y^*(x))v - \nabla^2_{21} f(x,y^*(x))v^*(x)\Big] \ .\end{aligned}$$

Let us consider $(v,x) \in \Gamma \times \mathbb{R}^d$ and $(v',x') \in \Gamma \times \mathbb{R}^d$. We have

$$\begin{aligned}\|\nabla_y \phi_v(v,x) - \nabla_y \phi_v(v',x')\| \leq &\|\nabla^2_{yy} g(x,y^*(x))v - \nabla^2_{yy} g(y^*(x'),x')v'\| \\ &+ \|\nabla_y f(x,y^*(x)) - \nabla_y f(y^*(x'),x')\|\end{aligned}$$

For the first term,

$$\begin{aligned}\|\nabla^2_{yy} g(x,y^*(x))v - \nabla^2_{yy} g(y^*(x'),x')v'\| &\leq \|\nabla^2_{yy} g(x,y^*(x))(v-v')\| \\ &\quad + \|(\nabla^2_{yy} g(x,y^*(x)) - \nabla^2_{yy} g(y^*(x'),x'))v'\| \\ &\quad + \|\nabla^2_{yy} g(y^*(x'),x')(v-v')\| \\ &\leq 2L_{g,1}\|v-v'\| + L_{g,2}(L_{y^*}+1)\|v'\|\|x-x'\| \\ &\leq [2L_{g,1} + L_{g,2}(L_{y^*}+1)R]\|(v,x)-(v',x')\|\end{aligned}$$

For the second terms, we use the smoothness of f and the Lipschitz continuity of y^* (Lemma B.1):

$$\begin{aligned}\|\nabla_y f(x,y^*(x)) - \nabla_y f(y^*(x'),x')\| &\leq L_{f,1}\|(x,y^*(x)) - (y^*(x'),x')\| \\ &\leq L_{f,1}(\|y^*(x) - y^*(x')\| + \|x-x'\|) \\ &\leq L_{f,1}(L_{y^*}+1)\|x-x'\| \\ &\leq L_{f,1}(L_{y^*}+1)\|(x,v)-(x',v')\| \ .\end{aligned}$$

As a consequence

$$\|\nabla_y \phi_v(v,x) - \nabla_y \phi_v(v',x')\| \leq \Lambda_1 \|(v,x)-(v',x')\| \tag{B.1}$$

with

$$\Lambda_1 = L_{f,1}(L_{y^*}+1) + 2L_{g,1} + L_{g,2}(L_{y^*}+1)R \ . \tag{B.2}$$

To prove the Lipschitz continuity of $\nabla_x \phi_v$, we remark that $\nabla^3_{yyy} g$, $\nabla^3_{xyy} g$ are Lipschitz and bounded by assumption. $(v \mapsto v)$ is Lipschitz and bounded on Γ. Also by Lemma B.1, y^* and v^* are Lipschitz and bounded. Finally, $\mathrm{d}y^*$ is bounded (Lemma B.1) and Lipschitz according to Chen et al. (2021)[Lemma 9]. As a consequence, $\nabla_x \phi_v$ is Λ_2-Lpischitz for some constant $\Lambda_2 > 0$. Hence, $\nabla \phi_v$ is Λ_v-Lipschitz continuous with $\Lambda_v = \Lambda_1 + \Lambda_2$.

□

Lemma B.5. *Let $t > 0$. For $k \in [q-1]$, we have*

$$0 \leq -\left\langle \frac{1}{\rho}(v^{t,k+1} - v^{t,k}) + D_v^{t,k}, v^{t,k+1} - v^{t,k} \right\rangle$$

Proof. The function ι_Γ being convex (since Γ is convex), let us consider its sub-differential

$$\partial \iota_\gamma(v) = \{\eta \in \mathbb{R}^p, \forall v' \in \mathbb{R}^p, \iota_\Gamma(v') \geq \iota_\Gamma(v) + \langle \eta, v'-v\rangle\}$$

By definition

$$v^{t,k+1} = \arg\min_v (\iota_\Gamma(v) + \frac{1}{2\rho}\|v - (v^{t,k} - \rho D_v^{t,k})\|^2) \ .$$

Using Fermat's rule, we get

$$-\frac{1}{\rho}(v^{t,k+1}-v^{t,k})-D_v^{t,k}\in\partial\iota_\Gamma(v^{t,k+1}) \ .$$

We can use the definition of the sub-differential with $\eta=-\frac{1}{\rho}(v^{t,k+1}-v^{t,k})-D_v^{t,k}$ to get

$$\underbrace{\iota_\Gamma(v^{t,k+1})}_{=0}\leq\underbrace{\iota_\Gamma(v^{t,k})}_{=0}-\left\langle\frac{1}{\rho}(v^{t,k+1}-v^{t,k})+D_v^{t,k},v^{t,k+1}-v^{t,k}\right\rangle \ .$$

□

We can now turn to the proof of Lemma 5.2.

Proof. The smoothness of ϕ_v provides us the following upper bound

$$\begin{aligned}\phi_v(v^{t,k+1},x^{t,k+1})\leq\phi_v(v^{t,k},x^{t,k})&+\langle\Pi_\Gamma(v^{t,k}-\rho D_v^{t,k})-v^{t,k},D_v(y^*(x^{t,k}),v^{t,k},x^{t,k})\rangle\\&+\frac{\Lambda_v}{2}\rho^2\|\Pi_\Gamma(v^{t,k}-\rho D_v^{t,k})-v^{t,k}\|^2\\&-\gamma\langle D_x^{t,k},\nabla_x\phi_v(v^{t,k},x^{t,k})\rangle+\frac{\Lambda_v}{2}\gamma^2\|D_x^{t,k}\|^2 \ .\end{aligned}\tag{B.3}$$

Let us denote $\Delta_\Pi^{t,k}=\Pi_\Gamma(v^{t,k}-\rho D_v^{t,k})-\Pi_\Gamma(v^{t,k}-\rho D_v(y^*(x^{t,k}),v^{t,k},x^{t,k}))$. Adding and subtracting

$$\begin{aligned}&\langle\Pi_\Gamma(v^{t,k}-\rho D_v(y^*(x^{t,k}),v^{t,k},x^{t,k})-v^{t,k},D_v(y^*(x^{t,k}),v^{t,k},x^{t,k})\rangle\\&\quad+\frac{\Lambda_v}{2}\|\Pi_\Gamma(v^{t,k}-\rho D_v(y^*(x^{t,k}),v^{t,k},x^{t,k})-v^{t,k}\|^2\end{aligned}$$

yields

$$\begin{aligned}\phi_v(v^{t,k+1},x^{t,k+1})\leq\phi_v(v^{t,k},x^{t,k})&+\langle\Delta_\Pi^{t,k},D_v(y^*(x^{t,k}),v^{t,k},x^{t,k})\rangle\\&+\frac{\Lambda_v}{2}\|\Pi_\Gamma(v^{t,k}-\rho D_v(y^*(x^t),v^{t,k},x^{t,k}))-v^{t,k}\|^2\\&+\langle\Pi_\Gamma(v^{t,k}-\rho D_v(y^*(x^{t,k}),v^{t,k},x^{t,k}))-v^{t,k},D_v(y^*(x^{t,k}),v^{t,k},x^{t,k})\rangle\\&+\frac{\Lambda_v}{2}\|\Delta_\Pi^{t,k}\|^2+\Lambda_v\langle\Delta_\Pi^{t,k},\Pi_\Gamma(v^{t,k}-\rho D_v(y^*(x^{t,k}),v^{t,k},x^{t,k}))-v^{t,k}\rangle\\&-\gamma\langle D_x^{t,k},\nabla_x\phi_v(v^{t,k},x^{t,k})\rangle+\frac{\Lambda_v}{2}\gamma^2\|D_x^{t,k}\|^2 \ .\end{aligned}\tag{B.4}$$

Taking $\rho\leq\frac{1}{\Gamma_v}$ gives

$$\begin{aligned}\phi_v(v^{t,k+1},x^{t,k+1})\leq\phi_v(v^{t,k},x^{t,k})&+\langle\Delta_\Pi^{t,k},D_v(y^*(x^{t,k}),v^{t,k},x^{t,k})\rangle\\&+\frac{1}{2\rho}\|\Pi_\Gamma(v^{t,k}-\rho D_v(y^*(x^t),v^{t,k},x^{t,k}))-v^{t,k}\|^2\\&+\langle\Pi_\Gamma(v^{t,k}-\rho D_v(y^*(x^{t,k}),v^{t,k},x^{t,k}))-v^{t,k},D_v(y^*(x^{t,k}),v^{t,k},x^{t,k})\rangle\\&+\frac{\Lambda_v}{2}\|\Delta_\Pi^{t,k}\|^2+\Lambda_v\langle\Delta_\Pi^{t,k},\Pi_\Gamma(v^{t,k}-\rho D_v(y^*(x^{t,k}),v^{t,k},x^{t,k}))-v^{t,k}\rangle\\&-\gamma\langle D_x^{t,k},\nabla_x\phi_v(v^{t,k},x^{t,k})\rangle+\frac{\Lambda_v}{2}\gamma^2\|D_x^{t,k}\|^2 \ .\end{aligned}\tag{B.5}$$

Let ι_Γ the indicator function of the convex set Γ. Similarly to Karimi et al. (2016, Equation 13) we define for any $\alpha>0$ and $v\in\mathbb{R}^p$

$$\mathcal{D}_{\iota_\Gamma}(v,x,\alpha)=-2\alpha\min_{v'\in\mathbb{R}^p}\left[\langle\nabla_y\phi_v(v,x),v'-v\rangle+\frac{\alpha}{2}\|v'-v\|^2+\iota_\Gamma(v')-\iota_\Gamma(v)\right] \ .$$

Hence, for $v \in \Gamma$ and $x \in \mathbb{R}^d$, we have

$$-\frac{\rho}{2}\mathcal{D}_{\iota_\Gamma}\left(v, x, \frac{1}{\rho}\right) = \langle \Pi_\Gamma(v - \rho D_v(y^*(x), v, x)) - v, D_v(y^*(x), v, x)\rangle + \frac{1}{2\rho}\|\Pi_\Gamma(v - \rho D_v(y^*(x), v, x)) - v\|^2 \ .$$

Therefore, Equation (B.5) can be written as

$$\begin{aligned}\phi_v(v^{t,k+1}, x^{t,k+1}) \leq \phi_v(v^{t,k}, x^{t,k}) &- \frac{\rho}{2}\mathcal{D}_{\iota_\Gamma}\left(v^{t,k}, x^{t,k}, \frac{1}{\rho}\right) \\ &+ \langle \Delta_\Pi^{t,k}, D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\rangle \\ &+ \frac{\Lambda_v}{2}\|\Delta_\Pi^{t,k}\|^2 + \Lambda_v\langle \Delta_\Pi^{t,k}, \Pi_\Gamma(v^{t,k} - \rho D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})) - v^{t,k}\rangle \\ &- \gamma\langle D_x^{t,k}, \nabla_x \phi_v(v^{t,k}, x^{t,k})\rangle + \frac{\Lambda_v}{2}\gamma^2\|D_x^{t,k}\|^2 \ .\end{aligned}$$

By strong convexity of ϕ_v with respect top v and smoothness, we have

$$\mathcal{D}_{\iota_\Gamma}(v^{t,k}, x^{t,k}, \Lambda_v) \ \geq \ 2\mu_g \phi_v(v^{t,k}, x^{t,k}) \ .$$

According to Karimi et al. (2016, Lemma 1), $\mathcal{D}_{\iota_\Gamma}(v^{t,k}, x^{t,k}, \cdot)$ is an increasing function. As a consequence, since $\Lambda_v \leq \frac{1}{\rho}$, we have $\mathcal{D}_{\iota_\Gamma}\left(v^{t,k}, x^{t,k}, \frac{1}{\rho}\right) \geq 2\mu_g\phi_v(v^{t,k}, x^{t,k})$. This leads to

$$\begin{aligned}\phi_v(v^{t,k+1}, x^{t,k+1}) \leq (1 - \rho\mu_g)\phi_v(v^{t,k}, x^{t,k}) &+ \langle \Delta_\Pi^{t,k}, D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\rangle \\ &+ \frac{\Lambda_v}{2}\|\Delta_\Pi^{t,k}\|^2 + \Lambda_v\langle \Delta_\Pi^{t,k}, \Pi_\Gamma(v^{t,k} - \rho D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})) - v^{t,k}\rangle \\ &- \gamma\langle D_x^{t,k}, \nabla_x \phi_v(v^{t,k}, x^{t,k})\rangle + \frac{\Lambda_v}{2}\gamma^2\|D_x^{t,k}\|^2 \ .\end{aligned} \tag{B.6}$$

The non-expansiveness of Π_Γ yields

$$\|\Delta_\Pi^{t,k}\| \leq \rho\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\| \tag{B.7}$$

and

$$\begin{aligned}\|\Pi_\Gamma(v^{t,k} - \rho D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k}) - \underbrace{v^{t,k}}_{\in\Gamma}\| &= \|\Pi_\Gamma(v^{t,k} - \rho D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})) - \Pi_\Gamma(v^{t,k})\| \\ &\leq \rho\|D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\| \ .\end{aligned} \tag{B.8}$$

Moreover, using Equation (B.7) and Young Inequality, we have for any $c > 0$

$$\begin{aligned}\langle \Delta_\Pi^{t,k}, D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\rangle &\leq \frac{c}{2}\|\Delta_\Pi\|^2 + \frac{1}{2c}\|D_v(y^*(x^{t,k}), v^{t,k}, x^{tk,})\|^2 \\ &\leq \frac{c\rho^2}{2}\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\ &\quad + \frac{1}{2c}\|D_v(y^*(x^{t,k}), v^{t,k}, x^{tk,}) - \underbrace{D_v(y^*(x^{t,k}), v^*(x^{t,k}), x^{tk,})}_{=0}\|^2 \\ &\leq \frac{c\rho^2}{2}\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\ &\quad + \frac{L_{g,1}}{\mu_g c}\phi_v(v^{t,k}, x^{t,k})\end{aligned} \tag{B.9}$$

Plugging Equation (B.9) into Equation (B.6) with $c = \frac{2L_{g,1}}{\mu_g^2\rho}$ yields

$$\begin{aligned}\phi_v(v^{t,k+1}, x^{t,k+1}) \leq \left(1 - \frac{\rho\mu_g}{2}\right)\phi_v(v^{t,k}, x^{t,k}) &+ \frac{L_{g,1}\rho}{\mu_g^2}\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\ &+ \frac{\Lambda_v}{2}\|\Delta_\Pi^{t,k}\|^2 + \Lambda_v\langle \Delta_\Pi^{t,k}, \Pi_\Gamma(v^{t,k} - \rho D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})) - v^{t,k}\rangle \\ &- \gamma\langle D_x^{t,k}, \nabla_x \phi_v(v^{t,k}, x^{t,k})\rangle + \frac{\Lambda_v}{2}\gamma^2\|D_x^{t,k}\|^2 \ .\end{aligned} \tag{B.10}$$

Using Equation (B.7), Equation (B.8) and Young Inequality for $d > 0$ yields

$$\begin{aligned}\langle \Delta_{\Pi}^{t,k}, \Pi_{\Gamma}(v^{t,k} - \rho D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})) - v^{t,k} \rangle &\leq \frac{d}{2}\|\Delta_{\Pi}^{t,k}\|^2 \\ &\quad + \frac{1}{2d}\|\Pi_{\Gamma}(v^{t,k} - \rho D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})) - v^{t,k}\|^2 \\ &\leq \frac{d\rho^2}{2}\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\ &\quad + \frac{\rho^2}{2d}\|D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\ &\leq \frac{d\rho^2}{2}\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\ &\quad + \frac{L_1^G \rho^2}{\mu_g d}\phi_v(v^{t,k}, x^{t,k}) \ . \end{aligned} \tag{B.11}$$

(B.12)

Plugging Equation (B.12) into Equation (B.10) with $d = \frac{4L_{g,1}\Lambda_v)\rho}{\mu_g^2}$ gives

$$\begin{aligned}\phi_v(v^{t,k+1}, x^{t,k+1}) \leq &\left(1 - \frac{\rho\mu_g}{4}\right)\phi_v(v^{t,k}, x^{t,k}) \\ &+ \left[\frac{L_{g,1}\rho}{\mu_g^2} + \frac{2L_{g,1}\Lambda_v^2\rho^3}{\mu_g^2}\right]\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\ &+ \frac{\Lambda_v}{2}\|\Delta_{\Pi}^{t,k}\|^2 \\ &- \gamma\langle D_x^{t,k}, \nabla_x\phi_v(v^{t,k}, x^{t,k})\rangle + \frac{\Lambda_v}{2}\gamma^2\|D_x^{t,k}\|^2 \ . \end{aligned}$$

Using once again (B.7), we get

$$\begin{aligned}\phi_v(v^{t,k+1}, x^{t,k+1}) \leq &\left(1 - \frac{\rho\mu_g}{4}\right)\phi_v(v^{t,k}, x^{t,k}) \\ &+ \left[\frac{L_{g,1}\rho}{\mu_g^2} + \frac{2L_{g,1}\Lambda_v^2\rho^3}{\mu_g^2} + \frac{\Lambda_v\rho^2}{2}\right]\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\ &- \gamma\langle D_x^{t,k}, \nabla_x\phi_v(v^{t,k}, x^{t,k})\rangle + \frac{\Lambda_v}{2}\gamma^2\|D_x^{t,k}\|^2 \ . \end{aligned} \tag{B.13}$$

By Lemma B.5, we have for any $\alpha > 0$

$$0 \leq -\alpha\left\langle \frac{1}{\rho}(v^{t,k+1} - v^{t,k}) + D_v^{t,k}, v^{t,k+1} - v^{t,k} \right\rangle \ .$$

By adding this to Equation (B.13), we get

$$\begin{aligned}\phi_v(v^{t,k+1}, x^{t,k+1}) \leq &\left(1 - \frac{\rho\mu_g}{4}\right)\phi_v(v^{t,k}, x^{t,k}) \\ &- \frac{\alpha}{\rho}\|v^{t,k+1} - v^{t,k}\|^2 - \alpha\langle D_v^{t,k}, v^{t,k+1} - v^{t,k}\rangle \\ &+ \left[\frac{L_{g,1}\rho}{\mu_g^2} + \frac{2L_{g,1}\Lambda_v^2\rho^3}{\mu_g^2} + \frac{\Lambda_v\rho^2}{2}\right]\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\ &- \gamma\langle D_x^{t,k}, \nabla_x\phi_v(v^{t,k}, x^{t,k})\rangle + \frac{\Lambda_v}{2}\gamma^2\|D_x^{t,k}\|^2 \ . \end{aligned} \tag{B.14}$$

We can control $-\left\langle D_v^{t,k}, v^{t,k+1} - v^{t,k}\right\rangle$ by Cauchy-Schwarz and Young for some $c, d, e, f > 0$

$$
\begin{aligned}
-\left\langle D_v^{t,k}, v^{t,k+1} - v^{t,k}\right\rangle &= -\left\langle D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k}), \Pi_\Gamma(v^{t,k} - \rho D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k}) - v^{t,k}\right\rangle \\
&\quad - \left\langle D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k}), \Delta_\Pi^{t,k}\right\rangle \\
&\quad - \left\langle D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k}), \Pi_\Gamma(v^{t,k} - \rho D_v(y^*(x))) - v^{t,k}\right\rangle \\
&\quad - \left\langle D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k}), \Delta_\Pi^{t,k}\right\rangle \\
&\leq \frac{c}{2}\|D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\
&\quad + \frac{1}{2c}\|\Pi_\Gamma(v^{t,k} - \rho D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})) - v^{t,k}\|^2 \\
&\quad + \frac{d}{2}\|D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 + \frac{1}{2d}\|\Delta_\Pi^{t,k}\|^2 \\
&\quad + \frac{e}{2}\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\
&\quad + \frac{1}{2e}\|\Pi_\Gamma(v^{t,k} - \rho D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})) - v^{t,k}\|^2 \\
&\quad + \frac{f}{2}\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 + \frac{1}{2f}\|\Delta_\Pi^{t,k}\|^2 \\
&\leq \left(\frac{c+d}{2} + \rho^2\left(\frac{1}{2c} + \frac{1}{2e}\right)\right)\|D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\
&\quad + \left(\frac{e+f}{2} + \rho^2\left(\frac{1}{2d} + \frac{1}{2f}\right)\right)\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\
&\leq \left(\frac{c+d}{2} + \rho^2\left(\frac{1}{2c} + \frac{1}{2e}\right)\right)\frac{2L_{g,1}}{\mu_g}\phi_v(v^{t,k}, x^{t,k}) \\
&\quad + \left(\frac{e+f}{2} + \rho^2\left(\frac{1}{2d} + \frac{1}{2f}\right)\right)\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2
\end{aligned}
$$

Let us take $c = d = e = f = \rho$. We get

$$
-\left\langle D_v^{t,k}, v^{t,k+1} - v^{t,k}\right\rangle \leq \frac{4L_{g,1}}{\mu_g}\rho\phi_v(v^{t,k}, x^{t,k}) + 2\rho\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \ .
$$

Then, by plugging the last Inequality in Equation (B.14) and setting $\alpha = \frac{\mu_g^2}{32L_{g,1}}$, we end up with

$$
\begin{aligned}
\phi_v(v^{t,k+1}, x^{t,k+1}) &\leq \left(1 - \frac{\mu_g}{8}\rho\right)\phi_v(v^{t,k}, x^{t,k}) - \frac{\alpha}{\rho}\|v^{t,k+1} - v^{t,k}\|^2 \\
&\quad + \rho\left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}} + \frac{\Lambda_v\rho}{2} + \frac{2L_{g,1}\Lambda_v^2\rho^2}{\mu_g^2}\right]\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\
&\quad - \gamma\langle D_x^{t,k}, \nabla_x\phi_v(v^{t,k}, x^{t,k})\rangle + \frac{\Lambda_v}{2}\gamma^2\|D_x^{t,k}\|^2 \\
&\leq \left(1 - \frac{\mu_g}{8}\rho\right)\phi_v(v^{t,k}, x^{t,k}) - \frac{\mu_g^2}{32L_{g,1}}\rho\|\mathcal{G}_v^{t,k}\|^2 \\
&\quad + \rho\left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}} + \frac{\Lambda_v\rho}{2} + \frac{2L_{g,1}\Lambda_v^2\rho^2}{\mu_g^2}\right]\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\
&\quad - \gamma\langle D_x^{t,k}, \nabla_x\phi_v(v^{t,k}, x^{t,k})\rangle + \frac{\Lambda_v}{2}\gamma^2\|D_x^{t,k}\|^2 \ .
\end{aligned}
$$

Since $\rho \leq B_v \triangleq \left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right] \min\left(\frac{2}{\Lambda_v}, \frac{\mu_g}{\sqrt{2L_{g,1}}\Lambda_v}\right)$ yields

$$\begin{aligned}
\phi_v(v^{t,k+1}, x^{t,k+1}) &\leq \left(1 - \frac{\mu_g}{8}\rho\right)\phi_v(v^{t,k}, x^{t,k}) - \frac{\mu_g^2}{32L_{g,1}}\rho\|\mathcal{G}_v^{t,k}\|^2 \\
&\quad + 3\rho\left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right]\|D_v^{t,k} - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\
&\quad - \gamma\langle D_x^{t,k}, \nabla_x\phi_v(v^{t,k}, x^{t,k})\rangle + \frac{\Lambda_v}{2}\gamma^2\|D_x^{t,k}\|^2 \\
&\leq \left(1 - \frac{\mu_g}{8}\rho\right)\phi_v(v^{t,k}, x^{t,k}) - \frac{\mu_g^2}{32L_{g,1}}\rho\|\mathcal{G}_v^{t,k}\|^2 \\
&\quad + 6\rho\left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right]\|D_v^{t,k} - D_v(\mathbf{u}^{t,k})\|^2 \\
&\quad + 6\rho\left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right]\|D_v(\mathbf{u}^{t,k}) - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 \\
&\quad - \gamma\langle D_x^{t,k}, \nabla_x\phi_v(v^{t,k}, x^{t,k})\rangle + \frac{\Lambda_v}{2}\gamma^2\|D_x^{t,k}\|^2 \ .
\end{aligned}$$

Tacking the expectation conditionally to the past iterates yields

$$\begin{aligned}
\mathbb{E}_{t,k}[\phi_v(v^{t,k+1}, x^{t,k+1})] &\leq \left(1 - \frac{\mu_g}{8}\rho\right)\phi_v(v^{t,k}, x^{t,k}) - \frac{\mu_g^2}{32L_{g,1}}\rho\mathbb{E}_{t,k}[\|\mathcal{G}_v^{t,k}\|^2] \\
&\quad + 6\rho\left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right]\mathbb{E}_{t,k}[\|D_v^{t,k} - D_v(\mathbf{u}^{t,k})\|^2] \\
&\quad + 6\rho\left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right]\mathbb{E}_{t,k}[\|D_v(\mathbf{u}^{t,k}) - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2] \\
&\quad - \gamma\langle\mathbb{E}_{t,k}[D_x^{t,k}], \nabla_x\phi_v(v^{t,k}, x^{t,k})\rangle + \frac{\Lambda_v}{2}\gamma^2\mathbb{E}_{t,k}[\|D_x^{t,k}\|^2] \ .
\end{aligned} \tag{B.15}$$

From Young inequality, we have for any $c > 0$

$$\langle\mathbb{E}_{t,k}[D_x^{t,k}], \nabla_x\phi_v(v^{t,k}, x^{t,k})\rangle \leq c^{-1}\|\mathbb{E}_{t,k}[D_x^{t,k}]\|^2 + c\|\nabla_x\phi_v(v^{t,k}, x^{t,k})\|^2 \ . \tag{B.16}$$

Moreover, using the Lipschitz continuity of y^*, of $\nabla_{yy}^2 G$ and ∇_F and the fact that v and v^* are bounded,

we have

$$\begin{aligned}
\|\nabla_x \phi_v(v,x)\| &\leq \|\mathrm{d}z*(x)\| \left[\left\|\frac{1}{2}(\nabla^3_{111}g(x,y^*(x))|v)v - \frac{1}{2}(\nabla^3_{111}g(x,y^*(x))|v^*(x))v^*(x)\right\|\right. \\
&\quad \left. + \|\nabla^2_{yy}f(x,y^*(x))v - \nabla^2_{yy}f(x,y^*(x))v^*(x)\|\right] \\
&\quad + \|\frac{1}{2}(\nabla^3_{211}g(x,y^*(x))|v)v - \frac{1}{2}(\nabla^3_{211}g(x,y^*(x))|v^*(x))v^*(x)\| \\
&\quad + \|\nabla^2_{21}f(x,y^*(x))v - \nabla^2_{21}f(x,y^*(x))v^*(x)\| \\
&\leq L_* \left[\left\|\frac{1}{2}(\nabla^3_{111}g(x,y^*(x))|v-v^*(x))v - \frac{1}{2}(\nabla^3_{111}g(x,y^*(x))|v^*(x))(v-v^*(x))\right\|\right. \\
&\quad \left. + L_{f,2}\|v-v^*(x)\|\right] \\
&\quad + \left\|\frac{1}{2}(\nabla^3_{211}g(x,y^*(x))|v-v^*(x))v - \frac{1}{2}(\nabla^3_{211}g(x,y^*(x))|v^*(x))(v-v^*(x))\right\| \\
&\quad + L_{f,2}\|v-v^*(x)\| \\
&\leq L_* \left[\left\|\frac{1}{2}(\nabla^3_{111}g(x,y^*(x))|v-v^*(x))v\right\|\right. \\
&\quad \left. + \left\|\frac{1}{2}(\nabla^3_{111}g(x,y^*(x))|v^*(x))(v-v^*(x))\right\| + L_{f,2}\|v-v^*(x)\|\right] \\
&\quad + \left\|\frac{1}{2}(\nabla^3_{211}g(x,y^*(x))|v-v^*(x))v\right\| \\
&\quad + \left\|\frac{1}{2}(\nabla^3_{211}g(x,y^*(x))|v^*(x))(v-v^*(x))\right\| + L_{f,2}\|v-v^*(x)\| \\
&\leq L_* \left[\frac{L_{g,2}}{2}(\|v\| + \|v^*(x)\|)\|v-v^*(x))\| + L_{f,2}\|v-v^*(x)\|\right] \\
&\quad + \frac{L_{g,2}}{2}(\|v\| + \|v^*(x)\|)\|v-v^*(x))\| + L_{f,2}\|v-v^*(x)\| \\
&\leq (L_*+1)\left[L_{g,2}R + L_{f,2}\right]\|v-v^*(x)\| \ .
\end{aligned}$$

On the other hand, we have by strong convexity

$$\|v-v^*(x)\|^2 \leq \frac{2}{\mu_g}\phi_v(v,x) \ .$$

As a consequence, we have

$$\|\nabla_x \phi_v(v^{t,k},x^{t,k})\|^2 \leq L''\phi_v(v^{t,k},x^{t,k}) \tag{B.17}$$

with $L'' = \frac{2(L_*+1)^2[L_{g,2}R+L_{f,2}]^2}{\mu_g}$.

Plugging Inequalities (B.16) and (B.17) into (B.15) yields

$$\begin{aligned}
\mathbb{E}_{t,k}[\phi_v(v^{t,k+1},x^{t,k+1})] &\leq \left(1 - \frac{\mu_g}{8}\rho + cL''\gamma\right)\phi_v(v^{t,k},x^{t,k}) - \frac{\mu_g^2}{32L_{g,1}}\rho\mathbb{E}_{t,k}[\|\mathcal{G}_v^{t,k}\|^2] \\
&\quad + 6\rho\left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right]\mathbb{E}_{t,k}[\|D_v^{t,k} - D_v(\mathbf{u}^{t,k})\|^2] \\
&\quad + 6\rho\left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right]\mathbb{E}_{t,k}[\|D_v(\mathbf{u}^{t,k}) - D_v(y^*(x^{t,k}),v^{t,k},x^{t,k})\|^2] \\
&\quad + \frac{\gamma}{c}\|\mathbb{E}_{t,k}[D_x^{t,k}]\|^2 + \frac{\Lambda_v}{2}\gamma^2\mathbb{E}_{t,k}[\|D_x^{t,k}\|^2] \ .
\end{aligned}$$

The Lipschitz continuity of $\nabla^2_{yy}G$ and $\nabla_y F$ and the boundedness of v give us

$$\begin{aligned}\|D_v(\mathbf{u}^{t,k}) - D_v(y^*(x^{t,k}), v^{t,k}, x^{t,k})\|^2 &\leq \big(\|\nabla^2_{yy}g(y^{t,k}, x^{t,k})v^{t,k} - \nabla^2_{yy}g(y^*(x^{t,k}), x^{t,k})v^{t,k}\| \\ &\quad + \|\nabla_y f(y^{t,k}, x^{t,k}) - \nabla_y f(y^*(x^{t,k}), x^{t,k})\|\big)^2 \\ &\leq (L_{g,2}R + L_{f,1})^2\|y^{t,k} - y^*(x^{t,k})\|^2 \\ &\leq \frac{2(L_{g,2}R + L_{f,1})^2}{\mu_g}\phi_y(y^{t,k}, x^{t,k}) \ .\end{aligned}$$

As a consequence

$$\begin{aligned}\mathbb{E}_{t,k}[\phi_v(v^{t,k+1}, x^{t,k+1})] \leq \left(1 - \frac{\mu_g}{8}\rho + cL''\gamma\right)\phi_v(v^{t,k}, x^{t,k}) - \frac{\mu_g^2}{32L_{g,1}}\rho\mathbb{E}_{t,k}[\|\mathcal{G}_v^{t,k}\|^2] \\ + 6\rho\left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right]\mathbb{E}_{t,k}[\|D_v^{t,k} - D_v(\mathbf{u}^{t,k})\|^2] \\ + 6\rho\left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right]\frac{2(L_{g,2}R + L_{f,1})^2}{\mu_g}\phi_y(y^{t,k}, x^{t,k}) \\ + \frac{\gamma}{c}\|\mathbb{E}_{t,k}[D_x^{t,k}]\|^2 + \frac{\Lambda_v}{2}\gamma^2\mathbb{E}_{t,k}[\|D_x^{t,k}\|^2] \ .\end{aligned} \tag{B.18}$$

From Lemma B.2, we have

$$\begin{aligned}\mathbb{E}[\|D_v^{t,k} - D_v(\mathbf{u}^{t,k})\|^2] \leq 4\rho^2\left((L_{g,2}R)^2 + (L_{f,1})^2\right)\sum_{r=1}^{k}\mathbb{E}[\|D_y^{t,r-1}\|^2] + 4\rho^2(L_{g,1})^2\sum_{r=1}^{k}\mathbb{E}[\|\mathcal{G}_v^{t,r-1}\|^2] \\ + 4\gamma^2\left((L_{g,2}R)^2 + (L_{f,1})^2\right)\sum_{r=1}^{k}\mathbb{E}[\|D_x^{t,r-1}\|^2]\end{aligned}$$

Taking the total expectation and plugging the previous inequality in Equation (B.18) yields

$$\begin{aligned}\phi_v^{t,k+1} \leq \left(1 - \frac{\mu_g}{8}\rho + cL''\gamma\right)\phi_v^{t,k} - \frac{\mu_g^2}{32L_{g,1}}\rho\mathbb{E}_{t,k}[\|\mathcal{G}_v^{t,k}\|^2] \\ + 24\rho^3\left((L_{g,2}R)^2 + (L_{f,1})^2\right)\left(\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right)\sum_{r=1}^{k}\mathbb{E}[\|D_y^{t,r-1}\|^2] \\ + 24\rho^3(L_{g,1})^2\left(\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right)\sum_{r=1}^{k}\mathbb{E}[\|\mathcal{G}_v^{t,r-1}\|^2] \\ + 24\rho\gamma^2\left((L_{g,2}R)^2 + (L_{f,1})^2\right)\left(\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right)\sum_{r=1}^{k}\mathbb{E}[\|D_x^{t,r-1}\|^2] \\ + \left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right]\frac{12(L_{g,2}R + L_{f,1})^2}{\mu_g}\rho\phi_y^{t,k} \\ + \frac{\gamma}{c}\mathbb{E}[\|[\mathbb{E}_{t,k}D_x^{t,k}]\|^2] + \frac{\Lambda_v}{2}\gamma^2\mathbb{E}[\|D_x^{t,k}\|^2] \ .\end{aligned}$$

Taking $c = \frac{\mu_g \rho}{16 L'' \gamma}$ yields

$$\begin{aligned}
\phi_v^{t,k+1} \leq & \left(1 - \frac{\mu_g}{16}\rho\right) \phi_v^{t,k} - \frac{\mu_g^2}{32 L_{g,1}} \rho \mathbb{E}_{t,k}[\|\mathcal{G}_v^{t,k}\|^2] \\
& + 24\rho^3 \left((L_{g,2}R)^2 + (L_{f,1})^2\right) \left(\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16 L_{g,1}}\right) \sum_{r=1}^{k} \mathbb{E}[\|D_y^{t,r-1}\|^2] \\
& + 24\rho^3 (L_{g,1})^2 \left(\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16 L_{g,1}}\right) \sum_{r=1}^{k} \mathbb{E}[\|\mathcal{G}_v^{t,r-1}\|^2] \\
& + 24\rho\gamma^2 \left((L_{g,2}R)^2 + (L_{f,1})^2\right) \left(\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16 L_{g,1}}\right) \sum_{r=1}^{k} \mathbb{E}[\|D_x^{t,r-1}\|^2] \\
& + \left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16 L_{g,1}}\right] \frac{12 (L_{g,2}R + L_{f,1})^2}{\mu_g} \rho \phi_y^{t,k} \\
& + \frac{16 L''}{\mu_g} \frac{\gamma^2}{\rho} \mathbb{E}[\|[\mathbb{E}_{t,k} D_x^{t,k}]\|^2] + \frac{\Lambda_v}{2} \gamma^2 \mathbb{E}[\|D_x^{t,k}\|^2] \ .
\end{aligned}$$

Combining Equation (5.19) and Lemma B.2 yields

$$\begin{aligned}
\phi_v^{t,k+1} \leq & \left(1 - \frac{\mu_g}{16}\rho\right) \phi_v^{t,k} - \frac{\mu_g^2}{32 L_{g,1}} \rho \mathbb{E}_{t,k}[\|\mathcal{G}_v^{t,k}\|^2] \\
& + 8\rho \left((L_{g,2}R)^2 + (L_{f,1})^2\right) \left[3 \left(\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16 L_{g,1}}\right) \rho^2 + \frac{8 L''}{\mu_g} \gamma^2\right] \sum_{r=1}^{k} \mathbb{E}[\|D_y^{t,r-1}\|^2] \\
& + 8\rho (L_{g,1})^2 \left[3 \left(\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16 L_{g,1}}\right) \rho^2 + \frac{8 L''}{\mu_g} \gamma^2\right] \sum_{r=1}^{k} \mathbb{E}[\|\mathcal{G}_v^{t,r-1}\|^2] \\
& + 8\gamma^2 \left((L_{g,2}R)^2 + (L_{f,1})^2\right) \left[3 \left(\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16 L_{g,1}}\right) \gamma + \frac{8 L''}{\mu_g} \frac{\gamma^2}{\rho}\right] \sum_{r=1}^{k} \mathbb{E}[\|D_x^{t,r-1}\|^2] \\
& + \left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16 L_{g,1}}\right] \frac{12 (L_{g,2}R + L_{f,1})^2}{\mu_g} \rho \phi_y^{t,k} \\
& + \frac{16 L''}{\mu_g} \frac{\gamma^2}{\rho} \mathbb{E}[\|D_x(\mathbf{u}^{tk,})\|^2] + \frac{\Lambda_v}{2} \gamma^2 \mathbb{E}[\|D_x^{t,k}\|^2] \ .
\end{aligned}$$

By assumption, $\gamma \leq C_v \rho$ with $C_v = \sqrt{\frac{\mu_g}{8 L''} \left(\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16 L_{g,1}}\right)}$, therefore

$$\begin{aligned}
\phi_v^{t,k+1} \leq & \left(1 - \frac{\mu_g}{16}\rho\right) \phi_v^{t,k} - \frac{\mu_g^2}{32 L_{g,1}} \rho \mathbb{E}_{t,k}[\|\mathcal{G}_v^{t,k}\|^2] \\
& + 32\rho^3 \left((L_{g,2}R)^2 + (L_{f,1})^2\right) \left(\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16 L_{g,1}}\right) \sum_{r=1}^{k} \mathbb{E}[\|D_y^{t,r-1}\|^2] \\
& + 32\rho^3 (L_{g,1})^2 \left(\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16 L_{g,1}}\right) \sum_{r=1}^{k} \mathbb{E}[\|\mathcal{G}_v^{t,r-1}\|^2] \\
& + 32\gamma^2\rho \left((L_{g,2}R)^2 + (L_{f,1})^2\right) \left(\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16 L_{g,1}}\right) \sum_{r=1}^{k} \mathbb{E}[\|D_x^{t,r-1}\|^2] \\
& + \left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16 L_{g,1}}\right] \frac{12 (L_{g,2}R + L_{f,1})^2}{\mu_g} \rho \phi_y^{t,k} \\
& + \frac{16 L''}{\mu_g} \frac{\gamma^2}{\rho} \mathbb{E}[\|D_x(\mathbf{u}^{t,k})\|^2] + \frac{\Lambda_v}{2} \gamma^2 \mathbb{E}[\|D_x^{t,k}\|^2] \ .
\end{aligned}$$

We get finally

$$\phi_v^{t,k+1} \leq \left(1 - \frac{\rho\mu_g}{16}\right)\phi_v^{t,k} - \tilde{\beta}_{vv}\rho V_v^t + \rho^3\beta_{vy}\mathcal{V}_y^{t,k} + 2\rho^3\beta_{vv}\mathcal{V}_v^{t,k} + \gamma^2\rho\beta_{vx}\mathcal{V}_x^{t,k} + \rho\alpha_{vy}\phi_y^{t,k} + \frac{\Lambda_v}{2}\gamma^2\mathbb{E}[\|D_x^{t,k}\|^2] + \frac{\gamma^2}{\rho}\overline{\beta}_{vx}\mathbb{E}[\|D_x(\mathbf{u}^{t,k})\|^2]$$

with $\beta_{vy} = \beta_{vx} = 32\left((L_{g,2}R)^2 + (L_{f,1})^2\right)\left(\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right)$, $\beta_{vv} = (L_{g,1})^2\left(\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right)$, $\overline{\beta}_{vx} = \frac{16L''}{\mu_g}$, $\tilde{\beta}_{vv} = \frac{\mu_g^2}{32L_{g,1}}$
and $\alpha_{vy} = \left[\frac{L_{g,1}}{\mu_g^2} + \frac{\mu_g^2}{16L_{g,1}}\right]\frac{12(L_{g,2}R+L_{f,1})^2}{\mu_g}$. □

B.1.4 Proof of Theorem 5.1

The constants involved in Theorem 5.1 are

$$\psi_y = \frac{1}{16\overline{\beta}_{yx}}, \quad \psi_v = \min\left[\frac{1}{16\overline{\beta}_{vx}}, \frac{\alpha_{yv}\mu_g}{12}\psi_y\right]$$

$$\overline{\rho} = \min\left[\sqrt{\frac{\psi_y}{12q(\psi_y\beta_{yy} + \psi_v\beta_{yv})}}, \sqrt{\frac{1}{6\Lambda_y}}, \sqrt{\frac{1}{12q\beta_{vv}}}, B_v\right] ,$$

$$\xi = \min\left[C_y, C_v, 1, \frac{\psi_v\mu_g^2}{16L_x^2}, \sqrt{\frac{\mu_g}{8\overline{\beta}_{vx}}}, \frac{\psi_y\mu_g^2}{24L_x^2}, \sqrt{\frac{\mu_g}{12\overline{\beta}_{yx}}}\right] ,$$

$$\overline{\gamma} = \min\left[\sqrt{\frac{1}{12q(\psi_y\beta_{yx} + \psi_v\beta_{vx})}}, \sqrt{\frac{1}{12q\beta_{\Phi x}}}, \frac{1}{6(L_\Phi + \psi_y\Lambda_y + \psi_v\Lambda_v)}, \sqrt{\frac{\psi_v\tilde{\beta}_{vv}}{6q(\beta_{\Phi v} + \psi_y\beta_{vy})}}, \sqrt{\frac{\psi_y}{12q\beta_{\Phi y}}}\right] .$$

Proof. The proof is a classical Lyapunov analysis. Consider the following Lyapunov function $\mathcal{L}^{t,k} = \Phi^{t,k} + \psi_y\phi_y^{t,k} + \psi_v\phi_v^{t,k}$ for some positive constants ψ_y and ψ_v. We use use Lemmas 5.1 to 5.3 to upper bound $\mathcal{L}^{t,k} - \mathcal{L}^{t,k+1}$.

We have

$$\begin{aligned}\mathcal{L}^{t,k+1} - \mathcal{L}^{t,k} \leq & -\frac{\gamma}{2}g^{t,k} + (\psi_y\overline{\beta}_{yx} + \psi_v\overline{\beta}_{vx})\frac{\gamma^2}{\rho}\mathbb{E}[\|D_x(\mathbf{u}^{t,k})\|^2] \\ & + \left(\frac{2L_x^2}{\mu_g}\gamma - \psi_y\frac{\mu_g}{2}\rho + \psi_v\alpha_{yv}\rho\right)\phi_y^{t,k} + \left(\frac{2L_x^2}{\mu_g}\gamma - \psi_v\frac{\mu_g}{16}\rho\right)\phi_v^{t,k} \\ & + \left(\psi_y\frac{\Lambda_y}{2}\rho^2 - \psi_y\frac{1}{2}\rho\right)V_y^{t,k} - \psi_v\tilde{\beta}_{vv}\rho V_v^{t,k} \\ & + \left(\frac{L_\Phi}{2}\gamma^2 + \psi_y\frac{\Lambda_y}{2}\gamma^2 + \psi_v\frac{\Lambda_v}{2}\gamma^2 - \frac{\gamma}{2}\right)V_x^{t,k} \\ & + \left(\beta_{\Phi y}\rho\gamma^2 + \psi_y\beta_{yy}\rho^3 + \psi_v\beta_{yv}\rho^3\right)\mathcal{V}_y^{t,k} \\ & + \left(\beta_{\Phi v}\rho\gamma^2 + \psi_y\beta_{vy}\gamma^2\rho + \psi_v\beta_{vv}\rho^3\right)\mathcal{V}_v^{t,k} \\ & + \left(\beta_{\Phi x}\gamma^3 + \psi_y\beta_{yx}\gamma^2\rho + \psi_v\beta_{vx}\rho^3\right)\mathcal{V}_x^{t,k} .\end{aligned} \tag{B.19}$$

We bound $\mathbb{E}[\|D_x(\mathbf{u}^{t,k})\|^2]$ crudely by using Proposition 5.2

$$\begin{aligned}\mathbb{E}[\|D_x(\mathbf{u}^{t,k})\|^2] &\leq 2\mathbb{E}[\|\nabla\Phi(x^{t,k})\|^2] + 2\mathbb{E}[\|D_x(\mathbf{u}^{t,k}) - \nabla\Phi(x^{t,k})\|^2] \\ &\leq 2g^{t,k} + 2(\mathbb{E}[\|y^{t,k} - y^*(x^{t,k})\|^2] + \mathbb{E}[\|v^{t,k} - v^*(x^{t,k})\|^2]) \\ &\leq 2g^{t,k} + \frac{4}{\mu_g}(\phi_y^{t,k} + \phi_v^{t,k}) .\end{aligned}$$

Summing in (B.19) for $k = 0, \dots, q-1$ yields

$$\begin{aligned}
\mathcal{L}^{t,q} - \mathcal{L}^{t,0} \leq & -\left(\frac{\gamma}{2} - 2\psi_y\overline{\beta}_{yx}\frac{\gamma^2}{\rho} - 2\psi_v\overline{\beta}_{vx}\frac{\gamma^2}{\rho}\right)\sum_{k=0}^{q-1} g^{t,k} \\
& + \left(\frac{2L_x^2}{\mu_g}\gamma - \psi_y\frac{\mu_g}{2}\rho + \psi_v\alpha_{yv}\rho + \psi_y\overline{\beta}_{yx}\frac{\gamma^2}{\rho}\right)\sum_{k=0}^{q-1}\phi_y^{t,k} \\
& + \left(\frac{2L_x^2}{\mu_g}\gamma - \psi_v\frac{\mu_g}{16}\rho + \psi_v\overline{\beta}_{vx}\frac{\gamma^2}{\rho}\right)\sum_{k=0}^{q-1}\phi_v^{t,k} + \left(\psi_y\frac{\Lambda_y}{2}\rho^2 - \psi_y\frac{1}{2}\rho\right)\sum_{k=0}^{q-1} V_y^{t,k} \\
& - \psi_v\tilde{\beta}_{vv}\rho\sum_{k=0}^{q-1} V_v^{t,k} + \left(\frac{L_\Phi}{2}\gamma^2 + \psi_y\frac{\Lambda_y}{2}\gamma^2 + \psi_v\frac{\Lambda_v}{2}\gamma^2 - \frac{\gamma}{2}\right)\sum_{k=0}^{q-1} V_x^{t,k} \\
& + \left(\beta_{\Phi y}\rho\gamma^2 + \psi_y\beta_{yy}\rho^3 + \psi_v\beta_{yv}\rho^3\right)\sum_{k=0}^{q-1}\mathcal{V}_y^{t,k} \\
& + \left(\beta_{\Phi v}\rho\gamma^2 + \psi_y\beta_{vy}\gamma^2\rho + \psi_v\beta_{vv}\rho^3\right)\sum_{k=0}^{q-1}\mathcal{V}_v^{t,k} \\
& + \left(\beta_{\Phi x}\gamma^3 + \psi_y\beta_{yx}\gamma^2\rho + \psi_v\beta_{vx}\rho^3\right)\sum_{k=0}^{q-1}\mathcal{V}_x^{t,k} \ .
\end{aligned}$$

Since we have

$$\sum_{k=0}^{q-1}\mathcal{V}_\bullet^{t,k} = \sum_{k=0}^{q-1}\sum_{r=1}^{k}\mathbb{E}[\|D_\bullet^{t,r-1}\|^2] = \sum_{r=1}^{q-1}\sum_{k=r}^{q-1}\mathbb{E}[\|D_\bullet^{t,r-1}\|^2] = \sum_{r=1}^{q-1}(q-r)\mathbb{E}[\|D_\bullet^{t,r-1}\|^2] \leq q\sum_{k=1}^{q-1}\mathbb{E}[\|D_\bullet^{t,k-1}\|^2]$$

we get

$$\begin{aligned}
\mathcal{L}^{t,q} - \mathcal{L}^{t,0} \leq & -\left(\frac{\gamma}{2} - 2\psi_y\overline{\beta}_{yx}\frac{\gamma^2}{\rho} - 2\psi_v\overline{\beta}_{vx}\frac{\gamma^2}{\rho}\right)\sum_{k=0}^{q-1} g^{t,k} \\
& + \left(\frac{2L_x^2}{\mu_g}\gamma - \psi_y\frac{\mu_g}{2}\rho + \psi_v\alpha_{yv}\rho + \psi_y\overline{\beta}_{yx}\frac{\gamma^2}{\rho}\right)\sum_{k=0}^{q-1}\phi_y^{t,k} \\
& + \left(\frac{2L_x^2}{\mu_g}\gamma - \psi_v\frac{\mu_g}{2}\rho + \psi_v\overline{\beta}_{vx}\frac{\gamma^2}{\rho}\right)\sum_{k=0}^{q-1}\phi_v^{t,k} \\
& + \left(\psi_y\frac{\Lambda_y}{2}\rho^2 - \psi_y\frac{1}{2}\rho + q\left(\beta_{\Phi y}\rho\gamma^2 + \psi_y\beta_{yy}\rho^3 + \psi_v\beta_{yv}\rho^3\right)\right)\sum_{k=0}^{q-1} V_y^{t,k} \\
& + \left(-\psi_v\tilde{\beta}_{vv}\rho + q\left(\beta_{\Phi v}\rho\gamma^2 + \psi_y\beta_{vy}\gamma^2\rho + \psi_v\beta_{vv}\rho^3\right)\right)\sum_{k=0}^{q-1} V_v^{t,k} \\
& + \left(\left(\frac{L_\Phi}{2} + \psi_y\frac{\Lambda_y}{2} + \psi_v\frac{\Lambda_v}{2}\right)\gamma^2 - \frac{\gamma}{2} + q\left(\beta_{\Phi x}\gamma^3 + (\psi_y\beta_{yx} + \psi_v\beta_{vx})\rho\gamma^2\right)\right)\sum_{k=0}^{q-1} V_x^{t,k}.
\end{aligned} \tag{B.20}$$

Since $\rho \leq \overline{\rho} \leq \min\left[\sqrt{\frac{\psi_y}{12q(\psi_y\beta_{yy}+\psi_v\beta_{yv})}}, \sqrt{\frac{1}{6\Lambda_y}}\right]$ and $\gamma \leq \overline{\gamma} \leq \sqrt{\frac{\psi_y}{12q\beta_{\Phi y}}}$, we have

$$\psi_y\frac{\Lambda_y}{2}\rho^2 - \psi_y\frac{1}{2}\rho + q\left(\beta_{\Phi y}\rho\gamma^2 + \psi_y\beta_{yy}\rho^3 + \psi_v\beta_{yv}\rho^3\right) < 0 \ . \tag{B.21}$$

Moreover, the conditions $\rho \leq \overline{\rho} \leq \sqrt{\frac{\tilde{\beta}_{vv}}{6q\beta_{vv}}}$ and $\gamma \leq \overline{\gamma} \leq \sqrt{\frac{\psi_v\tilde{\beta}_{vv}}{6q(\beta_{\Phi v}+\psi_y\beta_{vy})}}$, ensure that

$$-\psi_v\tilde{\beta}_{vv}\rho + q\left(\beta_{\Phi v}\rho\gamma^2 + \psi_y\beta_{vy}\gamma^2\rho + \psi_v\beta_{vv}\rho^3\right) < 0 \ . \tag{B.22}$$

The conditions

$$\rho \leq \overline{\rho} \leq \sqrt{\frac{1}{12q(\psi_y \beta_{yx} + \psi_v \beta_{vx})}} \ ,$$

and

$$\gamma \leq \overline{\gamma} \leq \min \left[\sqrt{\frac{1}{12q(\psi_y \beta_{yx} + \psi_v \beta_{vx})}}, \sqrt{\frac{1}{12q\beta_{\Phi x}}}, \frac{1}{6(L_\Phi + \psi_y \Lambda_y + \psi_v \Lambda_v)} \right]$$

yield

$$\frac{L_\Phi}{2}\gamma^2 + \psi_y \frac{\Lambda_y}{2}\gamma^2 + \psi_v \frac{\Lambda_v}{2}\gamma^2 - \frac{\gamma}{2} + q\left(\beta_{\Phi x}\gamma^3 + \psi_y \beta_{yx}\gamma^2\rho + \psi_v \beta_{vx}\rho\gamma^2\right) < 0 \ . \tag{B.23}$$

The condition $\gamma \leq \xi\rho \leq \min \left[\frac{\psi_v \mu_g^2}{16L_x^2}, \sqrt{\frac{\mu_g}{8\overline{\beta}_{vx}}} \right] \rho$ ensures

$$\frac{2L_x^2}{\mu_g}\gamma - \psi_v \frac{\mu_g}{2}\rho + \psi_v \overline{\beta}_{vx}\frac{\gamma^2}{\rho} \leq 0 \tag{B.24}$$

By definition, we have $\psi_v \leq \frac{\alpha_{yv}\mu_g}{12}\psi_y$ and by assumptions $\gamma \leq \xi\rho \leq \min \left[\frac{\psi_y \mu_g^2}{24L_x^2}, \sqrt{\frac{\mu_g}{12\overline{\beta}_{yx}}} \right] \rho$. As a consequence

$$\frac{2L_x^2}{\mu_g}\gamma - \psi_y \frac{\mu_g}{2}\rho + \psi_v \alpha_{yv}\rho + \psi_y \overline{\beta}_{yx}\frac{\gamma^2}{\rho} < 0 \ . \tag{B.25}$$

Plugging Inequalities (B.21) to (B.25) into Equation (B.20) gives

$$\mathcal{L}^{t,q} - \mathcal{L}^{t,0} \leq -\left(\frac{\gamma}{2} - 2\psi_y \overline{\beta}_{yx}\frac{\gamma^2}{\rho} - 2\psi_v \overline{\beta}_{vx}\frac{\gamma^2}{\rho}\right) \sum_{k=0}^{q-1} g^{t,k} \ .$$

Since $\psi_y = \frac{\rho}{16\overline{\beta}_{yx}}$ and $\psi_v \leq \frac{\rho}{16\overline{\beta}_{vx}}$ and $\frac{\gamma^2}{\rho} \leq \xi \leq 1$, we get

$$\underbrace{\mathcal{L}^{t,q} - \mathcal{L}^{t,0}}_{\mathcal{L}^{t+1,0} - \mathcal{L}^{t,0}} \leq -\frac{\gamma}{4} \sum_{k=0}^{q-1} g^{t,k} \ .$$

Summing, telescoping and dividing by Tq gives

$$\frac{1}{Tq} \sum_{t=0}^{T-1} \sum_{k=0}^{q-1} g^{t,k} \leq \frac{4}{Tq\gamma} \underbrace{\left(\Phi^{0,0} - \Phi^* + \psi_y \phi^{0,0} + \psi_v \phi^{0,0}]\right)}_{\Gamma^0} \ .$$

□

B.2 Details on the experiments

We performed the experiments with the Python package Benchopt (Moreau et al., 2022)[1]. For each experiment, we use minibatches instead of single samples to estimate oracles because it is more efficient in practice. We use a batch size of 64 for the stochastic inner and outer oracles. All the experiments were performed on processors AMD EPYC 7742 (4 CPUs/experiment).

[1]The code of the benchmark is available at `\Phittps://github.com/benchopt/benchmark_bilevel` and the results are displayed in `https://benchopt.github.io/results/benchmark_bilevel.html`.

B.2.1 Benchmark on quadratics

For this benchmark, we consider

$$f(y,x)=\frac{1}{m}\sum_{j=1}^{m}F_j(y,x),\quad g(y,x)=\frac{1}{n}\sum_{i=1}^{n}g_i(y,x)\ .$$

The functions F_j and g_i are defined as

$$F_j(y,x)=\frac{1}{2}y^\top A_y^{F_j}z+\frac{1}{2}x^\top A_x^{F_j}+xB^{F_j}z+(d_y^{F_j})^\top z+(d_x^{F_j})^\top x$$
$$g_i(y,x)=\frac{1}{2}y^\top A_y^{g_i}z+\frac{1}{2}x^\top A_x^{g_i}+xB^{g_i}z+(d_y^{g_i})^\top z+(d_x^{g_i})^\top x$$

with $A_y^{F_j},A_y^{g_i}\in\mathbb{R}^{p\times p}$, $A_x^{F_j},A_x^{g_i}\in\mathbb{R}^{d\times d}$, $B^{F_j},B^{g_i}\in\mathbb{R}^{d\times p}$, $d_y^{F_j},d_y^{g_i}\in\mathbb{R}^p$ and $d_x^{F_j},d_x^{g_i}\in\mathbb{R}^d$. The vectors $d_x^{F_j},d_x^{g_i}$ are drawn randomly according to a normal distribution $\mathcal{N}(0,I_d)$. The vectors $d_y^{F_j},d_y^{g_i}$ are drawn randomly according to a normal distribution $\mathcal{N}(0,I_p)$. For the Hessian matrices with respect to z, we generate $A_y^{g_i}$ so that $\frac{1}{n}\sum_{i=1}^n A_y^{g_i}=A$ for a symetric positive definite matrix A with spectrum in $[0.1,1]$. To do so, we generate $x_i\sim\mathcal{N}(0,I_p)$ and set $A_y^{g_i}=\sqrt{A}x_i(\sqrt{A}x_i)^\top$. We proceed similarly for $A_y^{F_j}$, $A_y^{g_i}$, $A_x^{F_j}$. For B^{g_i}, we want $\frac{1}{n}\sum_{i=1}^n B^{g_i}=B$ for a prescribed matrix $B\in\mathbb{R}^{d\times p}$ such that $\|B\|=0.1$. Let $B=U\Sigma V^\top$ the singular values decomposition of B. To get B^{g_i}, we generate $x_i\sim\mathcal{N}(0,I_p)$ and set $B^{g_i}=(V\Sigma x_i)(Ux_i)^\top$. We proceed similarly for B^{F_j}. In our experiment, we take $n=32768$ and $m=1024$. To select the parameters of the solvers, we perform a grid search. More precisely, for each solver, we take the inner step size in the form of αt^{-a} where a is the theoretical decrease rate of each solver, and α is chosen in $\{0.01,0.1\}$. The outer step size is taken as $\frac{\alpha}{r}t^{-b}$ where b is the theoretical decrease rate and r is chosen in $\{0.1,1,10,100\}$. For the two-loops algorithms (*i.e.* StocBiO, VRBO, AmIGO), the number of inner steps is set to 10 after a manual search. In the methods implementing Neumann approximations (MRBO, VRBO, StocBiO), the number of terms in the Neumann series is also set to 10, and the scaling parameter η is set to 0.5. To get the fastest convergence, we keep for each solver the set of parameters that give the best decrease of Φ on the 100 first epochs. The period of full batch computation of VRBO and SRBA q is parametrized as $q=a\frac{n+m}{b}$ where $b=64$ is the batch size and a is chosen in $\{2^{-6},2^{-3},2^{-1},2^3,2^6\}$. For F^{2}SA, we take $\lambda_0=1$ and $\delta_t=\alpha t^{-\frac{1}{7}}$ with α chosen in $\{0.01,0.1,1\}$.

B.2.2 Hyperparameter selection with IJCNN1

We solve a regularization selection problem for an ℓ^2-regularized logistic regression problem. Here, we assume that we have a regularization parameter per feature. We have $n_{\text{train}}=49,990$ training samples $(d_i^{\text{train}},y_i^{\text{train}})_{i\in[n_{\text{train}}]}$ and $n_{\text{val}}=91,701$ validation samples $(d_i^{\text{val}},y_i^{\text{val}})_{i\in[n_{\text{train}}]}$ coming from the IJCNN1[2] dataset. Mathematically, it boils down to solve Problem (5.1) with f and g given by

$$f(\theta,\lambda)=\frac{1}{n_{\text{val}}}\sum_{j=1}^{n_{\text{val}}}\varphi(y_j^{\text{val}}\langle d_j^{\text{val}},\theta\rangle)$$
$$g(\theta,\lambda)=\frac{1}{n_{\text{train}}}\sum_{i=1}^{n_{\text{train}}}\varphi(y_i^{\text{train}}\langle d_i^{\text{train}},\theta\rangle)+\frac{1}{2}\sum_{k=1}^{p}\mathrm{e}^{\lambda_k}\theta_k^2$$

where φ is the logistic loss defined by $\varphi(u)=\log(1+\mathrm{e}^{-u})$. The inner and outer step sizes are set to 0.05.

To make our comparison, we select the parameters of each solver with an extensive grid search. More precisely, for each solver, we take the inner step size in the form of αt^{-a} where a is the theoretical decrease rate of each solver and α is chosen in $\{2^{-5},2^{-4},2^{-3},2^{-2}\}$. The outer step size is taken as $\frac{\alpha}{r}t^{-b}$ where b is the theoretical decrease rate and r is chosen in $\{10^{-2},10^{-1.5},10^{-1},10^{-0.5},10^0\}$. For

[2] 1https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

the two-loops algorithms (*i.e.* StocBiO, VRBO, AmIGO), the number of inner steps is set to 10 after a manual search. In the methods implementing Neumann approximations (MRBO, VRBO, StocBiO), the number of terms in the Neumann series is also set to 10, and the scaling parameter η is set to 0.5. To get the fastest convergence, we keep for each solver the set of parameters that give the best decrease of Φ on the 100 first epochs. The period of full batch computation of VRBO and SRBA q is parametrized as $q = a\frac{n+m}{b}$ where $b = 64$ is the batch size and a is chosen in $\{2^{-6}, 2^{-3}, 2^{-1}, 2^3, 2^6, 2^9\}$. For F^{2}SA, we take $\lambda_0 = 1$ and $\delta_t = \alpha t^{-\frac{1}{7}}$ with α chosen in $\{0.01, 0.1, 1\}$.

CHAPTER 6

CONCLUSION AND PERSPECTIVES

6.1 Conclusion

In this thesis, we investigated several aspects of stochastic AID-based algorithms. In particular, we extended some single-level optimization algorithms to the bilevel setting in the non-convex/strongly convex setting.

In Chapter 4, we tackled the challenge of stochastic bilevel optimization, exploring how established techniques for single-level optimization could be adapted to bilevel problems. We introduced a novel framework to easily adapt stochastic single-level optimization algorithms to bilevel problems when the outer and inner functions are defined as expectations. This is achieved by proposing dynamics that are linear in the outer and inner functions, facilitating the design of stochastic estimators for update directions. We proposed and analyzed two instantiations of this framework: SOBA, an adaptation of the stochastic gradient algorithm to the bilevel setting, and SABA, an adaptation of the variance-reduced algorithm SAGA. We demonstrated that those algorithms achieved complexity that is analogous to their single-level counterparts in nonconvex settings.

The complexity results of Chapter 4 led us to the question addressed in Chapter 5: does the complexity bound of single-level optimization transfer to bilevel optimization when the inner function is strongly convex? We considered the scenario where the outer and inner functions are finite sums and affirmed this by proving a lower bound on the complexity for solving such problems with algorithms based on stochastic approximate implicit differentiation. Additionally, we proposed an algorithm whose complexity matches this lower bound.

From a practical standpoint, we developed a comprehensive benchmark of bilevel optimization algorithms. This open-source benchmark includes a significant number of solvers covering a wide range of bilevel algorithms. It is designed to be enriched by the community, providing a reference for comparison of bilevel solvers and reducing the need to reimplement existing baselines.

6.2 Perspectives

Beyond the contributions of this thesis, several theoretical and practical questions remain open.

- **Memory-efficient warm-starting strategies.** In the introduction, we highlighted that a key element for the efficient application of AID-based algorithms is warm-starting the inner and iHVP sub-procedures. This process, however, requires storing the outputs of the subprocedures at

each outer iteration. When dealing with high-dimensional inner variables or multiple independent inner problems in a bilevel setting, this becomes impractical. Investigating memory-efficient strategies for warm-starting the sub-procedures, such as learning initializations (Sambharya et al., 2024), would be a valuable direction for future research.

- **Parameter free methods.** As in single-level optimization, bilevel optimization performance is highly dependent on step sizes. This is all the more difficult as in bilevel optimization, one might deal with the interaction between several dynamics. Only two papers have investigated this topic. Fan et al. (2024a) adapt Polyak step sizes to bilevel optimization (Polyak, 1987; Loizou et al., 2021). Salehi et al. (2023) proposes an inexact line search method for bilevel optimization, albeit requiring knowledge of the problem's regularity constants. Recently, much research has been done on parameter-free optimization methods with promising empirical results (Defazio and Mishchenko, 2023; Mishchenko and Defazio, 2024; Khaled et al., 2024; Schaipp et al., 2024). One possible direction would be to adapt these methods to the context of bilevel optimization.
- **Understanding AID with non-strongly convex inner functions.** Implicit differentiation assumes that the inner function is strongly convex. However, methods using approximate implicit differentiation have shown promising empirical results in deep learning contexts (Grangier et al., 2023). It would be interesting to investigate theoretically the performances of AID-based methods when dealing with a non-strongly convex inner function. What explains these practical performances? Is there a meaning of applying AID-based methods to bilevel problems with non-strongly convex inner functions?
- **Does gradient-based hyperparameter selection overfit the validation set?** The bilevel optimization framework is a natural formalization of this purpose. Moreover, this formalization enabled the development of efficient gradient-based methods to tune hyperparameters. While these methods can precisely solve bilevel problems, such precise fitting of the validation set might lead to overfitting. It would be beneficial to develop generalization bounds that quantify the risk of overfitting in gradient-based hyperparameter selection.

SUMMARY OF CONTRIBUTIONS

1 Context

Bilevel optimization is a subfield of optimization with more and more applications in machine learning. Among these applications, there are hyperparameter selection (Pedregosa, 2016; Franceschi et al., 2018), implicit deep learning (Bai et al., 2019), meta-learning (Rajeswaran et al., 2019), neural architecture search (Liu et al., 2019), and data augmentation (Cubuk et al., 2019; Rommel et al., 2022).

A bilevel optimization problem is an optimization problem in which some variables are constrained to be the solution of another optimization problem. Formally, consider f and g two real-valued functions defined on $\mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$. The bilevel problems we are interested in take the following form:

$$\min_{x \in \mathbb{R}^{d_x}} \Phi(x) \triangleq f(x, y^*(x)) \quad \text{s.t.} \quad y^*(x) \in \arg\min_{y \in \mathbb{R}^{d_y}} g(x, y) \ . \tag{6.1}$$

The function f is the *outer function*, and the function g is the *inner function*. Similarly, the variable x is the *outer variable*, and the variable y is the *inner variable*.

When the inner function $g(x, \cdot)$ is twice differentiable and strongly convex and the outer function f is differentiable, the Implicit Function Theorem enables to get the following expression of the gradient of Φ

$$\nabla \Phi(x) = \nabla_x f(x, y^*(x)) + \nabla^2_{xy} g(x, y^*(x)) v^*(x) \tag{6.2}$$

where $v^*(x)$ is the solution of a linear system

$$v^*(x) = -\left[\nabla^2_{yy} g(x, y^*(x))\right]^{-1} \nabla_y f(x, y^*(x)) \ .$$

The hypergradient given at Equation (6.2) presents two major computational bottlenecks: it requires the solutions of the inner problem $y^*(x)$ and of the linear system $v^*(x)$. When the inner problem is ill-conditioned, this can be dramatically costly.

Moreover, in many machine learning applications, the inner and the outer functions take the form of empirical means over a potentially large amount of samples

$$f(x, y) = \frac{1}{m} \sum_{j=1}^{n} f_j(x, y) \quad \text{and} \quad g(x, y) = \frac{1}{n} \sum_{i=1}^{m} g_i(x, y) \ . \tag{6.3}$$

When the numbers of samples m and n are large, evaluating the functions f and g and their derivatives can be costly. In single-level optimization, stochastic algorithms such as SGD (Robbins and Monro, 1951) suit this situation. This explains why we are interested in developing stochastic algorithms for bilevel problems that use only a handful of samples to progress.

2 Contributions

2.1 A framework for bilevel optimization that enables stochastic and global variance reduction algorithms

In Dagréou et al. (2022a), we propose a general framework to solve the Problem (6.1) without computing exactly the values of $y^*(x)$ and $v^*(x)$. The idea is to replace $y^*(x)$ and $v^*(x)$ in Equation (6.2) by auxiliary variables y and v that move simultaneously with x towards their respective equilibrium following suitable directions at iteration t:

$$y^{t+1} = y^t - \rho D_y(y^t, v^t, x^t) = y^t - \rho^t \nabla_y g(x^t, y^t) \tag{6.4}$$
$$v^{t+1} = v^t - \rho D_v(y^t, v^t, x^t) = v^t - \rho^t(\nabla^2_{yy} g(x^t, y^t)v^t + \nabla_y f(x^t, y^t)) \tag{6.5}$$
$$x^{t+1} = x^t - \gamma D_x(y^t, v^t, x^t) = x^t - \gamma^t(\nabla^2_{xy} g(x^t, y^t)v^t + \nabla_x f(x^t, y^t)) \tag{6.6}$$

where t is the iteration number and $\rho, \gamma > 0$ are step sizes. The advantage of considering the directions D_y, D_v, and D_x is that they are linear in the functions f and g. Therefore, when f and g are empirical means, these directions can be easily estimated by cheap stochastic estimators. Thus, our general framework summarized in Algorithm 7 consists of following the dynamics given by the equations (6.4), (6.5) and (6.6) where the directions $D_y(y^t, v^t, x^t)$, $D_v(y^t, v^t, x^t)$ and $D_x(y^t, v^t, x^t)$ are replaced by cheap stochastic estimators D^t_y, D^t_v and D^t_x.

Algorithm 7 General framework

Input: initialisations $y_0 \in \mathbb{R}^{d_y}$, $x_0 \in \mathbb{R}^{d_x}$, $v_0 \in \mathbb{R}^{d_y}$, number of iterations T, step size sequences $(\rho^t)_{t<T}$ and $(\gamma^t)_{t<T}$.
for $t = 0, \dots, T-1$ **do**
  Update y: $y^{t+1} = y^t - \rho^t D^t_y$,
  Update v: $v^{t+1} = v^t - \rho^t D^t_v$,
  Update x: $x^{t+1} = x^t - \gamma^t D^t_x$,
  where D^t_y, D^t_v and D^t_x are unbiased estimators of $D_y(y^t, v^t, x^t), D_v(y^t, v^t, v^t)$ and $D_x(y^t, v^t, x^t)$.
end for

We propose and analyze two instantiations of this general framework. The analysis of both instantiations relies on the following regularity assumptions that ensure that the value function Φ is differentiable with Lipschitz continuous gradient and that the directions D_y, D_v, and D_x are well-behaved.

Assumption 6.1. *The function f is twice differentiable. The derivatives ∇f and $\nabla^2 f$ are Lipschitz continuous in (x, y) with respective Lipschitz constants $L_{f,1}$ and $L_{f,2}$.*

Assumption 6.2. *The function g is three times continuously differentiable on $\mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$. For any $x \in \mathbb{R}^{d_x}$, $g(x, \cdot)$ is μ_g-strongly convex. The derivatives ∇g, $\nabla^2 g$ and $\nabla^3 g$ are Lipschitz continuous in with respective Lipschitz constants $L_{g,1}$, $L_{g,2}$ and $L_{g,3}$.*

Assumption 6.3. *There exists $C_f > 0$ such that for any x we have $\|\nabla_y f(x, y^*(x))\| \leq C_f$.*

The Lipschitz continuity of the derivatives up to the first order for f and the second order for g is standard in the literature (Ji et al., 2021; Arbel and Mairal, 2022a). However, to avoid assuming increasing batch size with the precision (Arbel and Mairal, 2022a), or avoid assuming the resolution of the linear system up to a precision ϵ in the analysis (Chen et al., 2021; Ji et al., 2021), we assume the Lipschitz continuity of the derivatives up to the second-order for f and the third-order for g.

We also make the following assumptions on the variance of the stochastic estimators.

Assumption 6.4. *There exists B_y, B_v and B_x such that for all t, $\mathbb{E}_t[\|D^t_y\|^2] \leq B^2_y(1 + \|D_y(y^t, v^t, x^t)\|^2)$ and $\mathbb{E}_t[\|D^t_v\|^2] \leq B^2_v(1 + \|D_v(y^t, v^t, x^t)\|^2)$ where $\mathbb{E}_t$ denotes the expectation conditionally to (y^t, v^t, x^t).*

Assumption 6.5. *There exists B_x such that for all t, $\mathbb{E}_t[\|D_x^t\|^2] \leq B_x^2$.*

StOchastic Bilevel Algorithm (SOBA). The first is SOBA, an adaptation of Stochastic Gradient Descent (SGD) (Robbins and Monro, 1951) for bilevel problems. It consists of taking for the directions

$$\begin{aligned} D_y^t &= \nabla_y g_i(x^t, y^t) \\ D_v^t &= \nabla_{yy}^2 g_i(x^t, y^t) v^t + \nabla_y f_j(x^t, y^t) \\ D_x^t &= \nabla_{xy}^2 g_i(x^t, y^t) v^t + \nabla_x f_j(x^t, y^t) \end{aligned}$$

for two randomly sampled indices $i \in [n]$ and $j \in [m]$.

We provide convergence guarantees for SOBA with fixed step sizes that depend on the horizon (Theorem 6.1). We show that SOBA achieves a convergence rate similar to the convergence rate of SGD for smooth non-convex single-level problems (Ghadimi and Lan, 2013) in terms of the dependency in T.

Theorem 6.1 (Convergence of SOBA, fixed step size)**.** *Fix an iteration $T > 1$ and assume that Assumptions 6.1 to 4.5 hold. We consider fixed steps $\rho^t = \Theta\left(\frac{1}{\sqrt{T}}\right)$ and $\gamma^t = \Theta\left(\frac{1}{\sqrt{T}}\right)$. Let $(x^t)_{t\geq 1}$ the sequence of outer iterates for SOBA. Then,*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|\nabla \Phi(x^t)\|^2] = \mathcal{O}(T^{-\frac{1}{2}}) \ .$$

In Theorem 6.2, we provide a convergence rate for SOBA with decreasing step sizes. Again, the rates we get are similar to the ones obtained for the SGD in the smooth non-convex single-level setting (Ghadimi and Lan, 2013).

Theorem 6.2 (Convergence of SOBA, decreasing step size)**.** *Assume that Assumptions 6.1 to 4.5 hold. We consider steps $\rho^t = \Theta\left(\frac{1}{\sqrt{t}}\right)$ and $\gamma^t = \Theta\left(\frac{1}{\sqrt{t}}\right)$. Let $(x^t)_{t\geq 1}$ the sequence of outer iterates for SOBA. Then,*

$$\inf_{t \leq T} \mathbb{E}[\|\nabla \Phi(x^t)\|^2] = \mathcal{O}(\log(T) T^{-\frac{1}{2}}) \ .$$

Stochastic Averaged Bilevel Algorithm (SABA). The second instantiation we propose is SABA, an adaptation of the variance-reduced algorithm SAGA (Defazio et al., 2014) for bilevel problems. The general idea is to replace each sum in the directions D by a sum over a memory, updating only one term at each iteration. To help with the exposition, we denote the vector of joint variables as $z = (y, x, v)$. Since we have sums over i and over j, we have two memories for each variable: w_i^t for $i \in [n]$ and $\tilde{w}_j^t$ for $j \in [m]$, which keep track of the previous values of the variable y. At each iteration t, we draw two random independent indices $i \in [n]$ and $j \in [m]$ uniformly and update the memories. To do so, we put $w_i^{t+1} = y^t$ and $w_{i'}^{t+1} = w_{i'}^t$ for $i' \neq i$, and $\tilde{w}_j^{t+1} = y^t$ and $\tilde{w}_{j'}^{t+1} = \tilde{w}_{j'}^t$ for $j' \neq j$. Each sum in the directions D is then approximated using SAGA-like rules: given n functions $\phi_{i'}$ for $i' \in [n]$, we define

$$S[\phi, w]_i^t = \phi_i(w_i^{t+1}) - \phi_i(w_i^t) + \frac{1}{n}\sum_{i'=1}^{n} \phi_{i'}(w_{i'}^t) \ .$$

This is an unbiased estimator of the average of the ϕ's since $\mathbb{E}_i\left[S[\phi, w]_i^t\right] = \frac{1}{n}\sum_{i=1}^n \phi_i(y^t)$.

With a slight abuse of notation, we call $\nabla_{yy}^2 gv$ the sequence of functions $(y \mapsto \nabla_{yy}^2 g_i(x, y)v)_{i\in[n]}$ and $\nabla_{xy}^2 gv$ the sequence of functions $(y \mapsto \nabla_{xy}^2 g_i(x, y)v)_{i\in[n]}$. Finally, the stochastic directions D_y^t, D_v^t and D_x^t used for the SABA algorithm are given by

$$\begin{aligned} D_y^t &= S[\nabla_y g, w]_i^t \\ D_v^t &= S[\nabla_{yy}^2 gv, w]_i^t + S[\nabla_y f, \tilde{w}]_j^t \\ D_x^t &= S[\nabla_{xy}^2 gv, w]_i^t + S[\nabla_x f, \tilde{w}]_j^t \ . \end{aligned}$$

Analogously to the SAGA's analysis, the SABA's analysis assumes that the regularity assumptions hold for each f_j and g_i instead of only the sums f and g.

Assumption 6.6. *For all $i \in [n]$ and $j \in [m]$, the functions ∇g_i, ∇f_j, $\nabla^2_{yy} g_i$ and $\nabla^2_{xy} g_i$ are Lipschitz continuous in (x, y).*

In Theorem 6.3, we show that SABA achieves a convergence rate in $\mathcal{O}((n+m)^{\frac{2}{3}}\epsilon^{-1})$. This rate is similar to the convergence rate of SAGA for non-convex problems (Reddi et al., 2016) in terms of the dependency on the number of samples and the iteration number.

Theorem 6.3 (Convergence of SABA, smooth case)**.** *Assume that Assumptions 6.1 to 6.3 and 6.6 hold. We suppose $\rho = \rho' N^{-\frac{2}{3}}$ and $\gamma = \xi\rho$, where ρ' and ξ depend only on regularity constants of f and g. Let $(x^t)_{t\geq 1}$ be the iterates of SABA. Then,*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|\nabla\Phi(x^t)\|^2] = O\left(N^{\frac{2}{3}}T^{-1}\right) \quad .$$

Finally, in Theorem 6.4, we show the linear convergence of SABA under a Polyak-Łojasiewicz (PL) assumption.

Theorem 6.4 (Convergence of SABA, PL case)**.** *Assume that Φ satisfies the PL inequality. This means that there exists a constant $\mu_\Phi > 0$ such that for any x we have $\frac{\mu_\Phi}{2}\|\Phi(x)\|^2 \geq \Phi(x) - \min_x \Phi(x)$. We furthermore assume that Assumptions 6.1 to 6.3 and 4.5 to 6.6 hold. We suppose $\rho = \rho' N^{-\frac{2}{3}}$ and $\gamma = \xi\rho' N^{-1}$, where ρ' and ξ depend only on f and g. Let x^t the iterates of SABA and $c' \triangleq \min\left(\mu_\Phi, \frac{1}{16P'}\right)$ with P' a constant. Then,*

$$\mathbb{E}[\Phi^T] - \Phi^* = (1 - c'\gamma)^T(\Phi^0 - \Phi^* + C^0)$$

where C^0 is a that depends on the initialization of y, v, x and memory.

2.2 A lower bound and a near-optimal algorithm for bilevel empirical risk minimization

We consider again Problem (6.1) in the finite sum setting given in Equation (6.3). In the previous, we show in two specific cases that bilevel algorithms adapted from single-level algorithms could have complexities similar to their single-level counterparts in the non-convex/strongly convex setting. This raises a natural question we try to answer in (Dagréou et al., 2024b): Do the lower and upper complexity bounds transfer from the single-level to the bilevel setting?

Upper bound. In the single-level setting, the algorithm SARAH (Nguyen et al., 2017) is known to be a near-optimal algorithm for the minimization of smooth non-convex finite sums (Nguyen et al., 2022). We propose an adaptation of SARAH for bilevel problems called SRBA.

Consider the dynamics given by Equations (6.4), (6.5) and (6.6). The SRBA algorithm consists of following these dynamics with stochastic estimators of the directions built recursively and reset periodically with a full batch computation of the directions. The algorithm is summarized in Algorithm 8 where $\Pi(y, v, x) = (y, \Pi_{B(0,R)}(v), x)$ with $\Pi_{B(0,R)}$ the projection on the ball of radius $R = \sup_{x\in\mathbb{R}^{d_x}} \|v^*(x)\|^2$ centered at 0. This projection enables avoiding a boundedness assumption on the iterates v.

The convergence analysis of SRBA relies on the following regularity assumptions. Note that, as opposed to the SOBA's analysis, our regularity assumptions should hold for each f_j and g_i instead of only the sums f and g.

Assumption 6.7. *For all $j \in [m]$, f_j is twice differentiable and $L_{f,0}$-Lipschitz continuous. Its gradient is $L_{f,1}$-Lipschitz continuous and its Hessian is $L_{f,2}$-Lipschitz continuous.*

Assumption 6.8. *For all $i \in [n]$, g_i is three times differentiable. Its first, second, and third order derivatives are respectively $L_{g,1}$-Lipschitz continuous, $L_{g,2}$-Lipschitz continuous, and $L_{g,3}$-Lipschitz continuous. For $x \in \mathbb{R}^d$, the function $g_i(x, \cdot)$ is μ_g-strongly convex.*

Based on these assumptions, we show the convergence rate of SRBA in Theorem 6.5.

Algorithm 8 Stochastic Recursive Bilevel Algorithm

Input: initializations $y_0 \in \mathbb{R}^p$, $x_0 \in \mathbb{R}^d$, $v_0 \in \mathbb{R}^p$, number of iterations T and q, step sizes ρ and γ.
Set $\tilde{\mathbf{u}}^{\mathbf{0}} = (y_0, v_0, x_0)$
for $t = 0, \dots, T-1$ **do**
  Reset $\boldsymbol{\Delta}$: $\boldsymbol{\Delta}^{t,0} = (\rho D_y(\tilde{\mathbf{u}}^t), \rho D_v(\tilde{\mathbf{u}}^t), \gamma D_x(\tilde{\mathbf{u}}^t))$
  Update $\mathbf{u}$: $\mathbf{u}^{t,1} = \Pi(\tilde{\mathbf{u}}^t - \boldsymbol{\Delta}^{\mathbf{t},\mathbf{0}})$,
  for $k = 1, \dots, q-1$ **do**
    Draw $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$
    $\boldsymbol{\Delta}_y^{t,k} = \rho(D_{y,i,j}(\mathbf{u}^{t,k}) - D_{y,i,j}(\mathbf{u}^{t,k-1})) + \boldsymbol{\Delta}_y^{t,k-1}$
    $\boldsymbol{\Delta}_v^{t,k} = \rho(D_{v,i,j}(\mathbf{u}^{t,k}) - D_{v,i,j}(\mathbf{u}^{t,k-1})) + \boldsymbol{\Delta}_v^{t,k-1}$
    $\boldsymbol{\Delta}_x^{t,k} = \gamma(D_{x,i,j}(\mathbf{u}^{t,k}) - D_{x,i,j}(\mathbf{u}^{t,k-1})) + \boldsymbol{\Delta}_x^{t,k-1}$
    Update $\mathbf{u}$: $\mathbf{u}^{t,k+1} = \Pi(\mathbf{u}^{t,k} - \boldsymbol{\Delta}^{t,k})$
  end for
  Set $\tilde{\mathbf{u}}^{t+1} = \mathbf{u}^{t+1,q}$
end for
Return $(\tilde{y}^T, \tilde{v}^T, \tilde{x}^T) = \tilde{\mathbf{u}}^T$

Theorem 6.5. *Assume that Assumptions 6.7 and 6.8 hold. Assume that the step sizes verify $\rho \leq \overline{\rho}$ and $\gamma \leq \min(\overline{\gamma}, \xi\rho)$ for some constants ξ, $\overline{\rho}$ and $\overline{\gamma}$. Then it holds*

$$\frac{1}{Tq}\sum_{t=0}^{T-1}\sum_{k=0}^{q-1}\mathbb{E}[\|\nabla\Phi(x^{t,k})\|^2] = \mathcal{O}\left(\frac{1}{qT\gamma}\right)$$

where $\mathcal{O}$ hides regularity constants that are independent from n and m.

Theorem 6.5 implies that SRBA find an ϵ-stationary point of Φ after at most $\mathcal{O}\left((n+m)^{\frac{1}{2}}\varepsilon^{-1} \vee (n+m)\right)$ calls to oracles, for appropriate choice of step sizes ρ, γ and inner loop length q.

Lower bound. We also establish a lower bound result on the number of oracles that are necessary to solve bilevel problems. For this, one needs to define the function and the algorithm classes we consider.

For the function class, we consider outer and inner functions that are finite sums with sufficient regularity to make the value function Φ differentiable. This class is formally defined in Definition 6.1.

Definition 6.1. Let n, m two positive integers, $L_{f,1}$ and μ_g two positive constants. The class of the smooth empirical risk minimization problems denoted by $\mathcal{C}^{L_{f,1},\mu_g}$ is the set of pairs of real-valued function families $((f_j)_{1\leq j\leq m}, (g_i)_{1\leq i\leq n})$ defined on $\mathbb{R}^p \times \mathbb{R}^d$ such that for all $j \in [m]$, f_j is $L_{f,1}$ smooth and for all $i \in [n]$, g_i is twice differentiable and μ_g-strongly convex.

The algorithm class we define in Definition 6.2 is a class that includes several stochastic AID-based algorithms of the literature.

Definition 6.2. Given initial points y^0, v^0, x^0, a *linear bilevel algorithm* $\mathcal{A}$ is a measurable mapping such that for any $((f_j)_{1\leq j\leq m}, (g_i)_{1\leq i\leq n}) \in \mathcal{C}^{L_{f,1},\mu_g}$, the output of $\mathcal{A}((f_j)_{1\leq j\leq m}, (g_i)_{1\leq i\leq n})$ is a sequence $\{(y^t, v^t, x^t, i_t, j_t)\}_{t\geq 0}$ of points (y^t, v^t, x^t) and random variables $i_t \in [n]$ and $j_t \in [m]$ such that for all

$t \geq 0$

$$
\begin{aligned}
y^{t+1} &\in y^0 + \mathrm{Span}\{\nabla_y g_{i_0}(y^0, x^0), \dots, \nabla_y g_{i_t}(y^t, x^t)\} \\
v^{t+1} &\in v^0 + \mathrm{Span}\{\nabla^2_{yy} g_{i_0}(y^0, x^0) v^0 + \nabla_y f_{j_0}(y^0, x^0), \\
&\qquad \dots, \nabla^2_{yy} g_{i_t}(y^t, x^t) v^t + \nabla_y f_{j_t}(y^t, x^t)\} \\
x^{t+1} &\in x^0 + \mathrm{Span}\{\nabla^2_{xy} g_{i_0}(y^0, x^0) v^0 + \nabla_x f_{j_0}(y^0, x^0), \\
&\qquad \dots, \nabla^2_{xy} g_{i_t}(y^t, x^t) v^t + \nabla_x f_{j_t}(y^t, x^t)\}.
\end{aligned}
$$

We can derive a lower complexity bound for bilevel problems with these ingredients. We state the result in Theorem 6.6.

Theorem 6.6. *For any linear bilevel algorithm $\mathcal{A}$, and any L^F, n, Δ, ε, p such that $\varepsilon \leq (\Delta L^F m^{-1})/10^3$, there exists a dimension $d = \mathcal{O}(\Delta \varepsilon^{-1} m^{\frac{1}{2}} L^F)$, an element $((f_j)_{1\leq j\leq m}, (g_i)_{1\leq i\leq n}) \in \mathcal{C}^{L_{f,1},\mu_g}$ such that the value function h defined as in (5.1) satisfies $\Phi(x^0) - \inf_{x\in\mathbb{R}^d} \Phi(x) \leq \Delta$ and in order to find $\hat{x} \in \mathbb{R}^d$ such that $\mathbb{E}[\|\nabla\Phi(\hat{x})\|^2] \leq \varepsilon$, $\mathcal{A}$ needs at least $\Omega(m^{\frac{1}{2}}\varepsilon^{-1})$ calls to oracles of the form (5.10).*

Rigorously speaking, the function classes considered for the upper and lower bounds are different does not match. Indeed, for the analysis of SRBA, we need to assume a bit more regularity on the functions f_j and g_i to control the dynamic of the variable v. Moreover, the lower bound result is a partial result since any dependency in the number of inner functions n appears.

2.3 Extensive benchmark of bilevel optimization algorithms

The literature on stochastic bilevel optimization is rich with many variants of algorithms that come with theoretical guarantees. However, assessing their practical performance is crucial to understanding the benefits and limitations of each method and reinforcing theoretical claims. Many bilevel optimization papers provide empirical results on classical tasks such as hyperparameter selection or data cleaning. However, the code is frequently unavailable, or if it is available, its documentation is incomplete, leading to a lack of reproducibility in the field.

For these reasons, we propose an extensive benchmark of bilevel optimization algorithms on several tasks. This benchmark has been made with the Python package Benchopt (Moreau et al., 2022), which is a tool for building standardized and collaborative benchmarks. The code is open source and available on GitHub [1]. Also, an HTML page displays the latest results of the benchmark[2].

In the benchmark, 17 solvers are implemented:

- **5 stochastic without variance reduction solvers.** AmIGO (Arbel and Mairal, 2022a), stocBiO (Ji et al., 2021), BSA (Ghadimi and Wang, 2018), TTSA (Hong et al., 2023), SOBA (Dagréou et al., 2022a);
- **6 stochastic solvers with variance reduction.** MRBO (Yang et al., 2021), VRBO (Yang et al., 2021), SUSTAIN (Khanduri et al., 2021), FSLA (Li et al., 2022), SABA (Dagréou et al., 2022a), SRBA (Dagréou et al., 2024b);
- **3 Hessian-free solvers.** BOME (Ye et al., 2022), F2SA (Kwon et al., 2023a), PZOBO (Sow et al., 2022),
- **2 full-batch solvers based on Jaxopt (Blondel et al., 2021).** Jaxopt-GD, Jaxopt-ITD;
- **A zero-order solver.** `Optuna` (Akiba et al., 2019).

We consider three tasks: a toy problem with quadratic functions, the hyperparameter selection problem for ℓ^2-regularized logistic regression with the IJCNN1 and COVTYPE datasets, and the data cleaning task with the MNIST dataset.

[1] the code of the benchmark is available at `https://github.com/benchopt/benchmark_bilevel`

[2] the results of the benchmark are available at `https://benchopt.github.io/results/benchmark_bilevel.html`

RÉSUMÉ DES CONTRIBUTIONS

1 Contexte

L'optimisation bi-niveaux est un sous-domaine de l'optimisation qui a plus en plus d'applications en apprentissage statistique. Parmi elles, on trouve la sélection d'hyperparamètres (Pedregosa, 2016; Franceschi et al., 2018), l'apprentissage profond implicite (Bai et al., 2019), le méta apprentissage (Rajeswaran et al., 2019), la recherche d'architecture neuronale (Liu et al., 2019) et l'augmentation de données (Cubuk et al., 2019; Rommel et al., 2022).

Un problème d'optimisation bi-niveaux est un problème d'optimisation dans lequel certaines variables sont contraintes à être la solution d'un autre problème d'optimisation. Formellement, considérons f et g comme deux fonctions à valeurs réelles définies sur $\mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$. Les problèmes bi-niveaux qui nous intéressent prennent la forme suivante :

$$\min_{x \in \mathbb{R}^{d_x}} \Phi(x) \triangleq f(x, y^*(x)) \quad \text{s.t.} \quad y^*(x) \in \arg\min_{y \in \mathbb{R}^{d_y}} g(x, y) \ . \tag{6.7}$$

La fonction f est la *fonction externe*, et la fonction g est la *fonction interne*. De même, la variable x est la *variable externe*, et la variable y est la *variable interne*.

Lorsque la fonction interne $g(x, \cdot)$ est deux fois différentiable et fortement convexe et que la fonction externe f est différentiable, le Théorème des Fonctions Implicites permet d'obtenir l'expression suivante du gradient de Φ

$$\nabla \Phi(x) = \nabla_x f(x, y^*(x)) + \nabla^2_{xy} g(x, y^*(x)) v^*(x) \tag{6.8}$$

où $v^*(x)$ est la solution d'un système linéaire

$$v^*(x) = -\left[\nabla^2_{yy} g(x, y^*(x))\right]^{-1} \nabla_y f(x, y^*(x)) \ .$$

L'hypergradient donné à l'Equation (6.8) présente deux principaux verrous computationnels : il nécessite la solution du problème interne $y^*(x)$ et celle du système linéaire $v^*(x)$. Lorsque le problème interne est mal conditionné, cela peut être extrêmement coûteux.

De plus, dans de nombreuses applications d'apprentissage automatique, les fonctions internes et externes prennent la forme de moyennes empiriques sur un nombre potentiellement grand d'échantillons

$$f(x, y) = \frac{1}{m} \sum_{j=1}^{n} f_j(x, y) \quad et \quad g(x, y) = \frac{1}{n} \sum_{i=1}^{m} g_i(x, y) \ . \tag{6.9}$$

Lorsque les nombres d'échantillons m et n sont grands, évaluer les fonctions f et g ainsi que leurs dérivées peut être coûteux. En optimisation à un seul niveau, des algorithmes stochastiques tels que la descente de gradient stochastique (SGD) (Robbins and Monro, 1951) sont parfaitement adaptés à cette situation. Cela explique pourquoi nous nous intéressons au développement d'algorithmes stochastiques pour les problèmes bi-niveaux qui utilisent seulement quelques d'échantillons pour progresser.

2 Contributions

2.1 Un cadre pour l'optimisation bi-niveaux qui permet les algorithmes de réduction de variance stochastiques et globaux

Dans Dagréou et al. (2022a), nous proposons un cadre général pour résoudre le Problème (6.7) sans calculer exactement $y^*(x)$ et $v^*(x)$. L'idée est de remplacer $y^*(x)$ et $v^*(x)$ dans l'Equation (6.8) par des variables auxiliaires y et v qui se déplacent simultanément avec x vers leur équilibre respectif en suivant des directions appropriées à l'itération t :

$$y^{t+1} = y^t - \rho D_y(y^t, v^t, x^t) = y^t - \rho^t \nabla_y g(x^t, y^t) \tag{6.10}$$
$$v^{t+1} = v^t - \rho D_v(y^t, v^t, x^t) = v^t - \rho^t(\nabla^2_{yy} g(x^t, y^t)v^t + \nabla_y f(x^t, y^t)) \tag{6.11}$$
$$x^{t+1} = x^t - \gamma D_x(y^t, v^t, x^t) = x^t - \gamma^t(\nabla^2_{xy} g(x^t, y^t)v^t + \nabla_x f(x^t, y^t)) \tag{6.12}$$

où t est le numéro de l'itération et ρ, $\gamma > 0$ sont des tailles de pas. L'avantage de considérer les directions D_y, D_v et D_x est qu'elles sont linéaires dans les fonctions f et g. Par conséquent, lorsque f et g sont des moyennes empiriques, ces directions peuvent être facilement estimées par des estimateurs stochastiques peu coûteux. Ainsi, notre cadre général résumé dans l'Algorithme 9 consiste à suivre les dynamiques données par les équations (6.10), (6.11) et (6.12) où les directions $D_y(y^t, v^t, x^t)$, $D_v(y^t, v^t, x^t)$ et $D_x(y^t, v^t, x^t)$ sont remplacées par des estimateurs stochastiques D^t_y, D^t_v et D^t_x.

Algorithm 9 Cadre général

Entrée : initialisations $y_0 \in \mathbb{R}^{d_y}$, $x_0 \in \mathbb{R}^{d_x}$, $v_0 \in \mathbb{R}^{d_y}$, nombre d'itérations T, séquences de tailles de pas $(\rho^t)_{t<T}$ et $(\gamma^t)_{t<T}$.
for $t = 0, \dots, T-1$ **do**
  Mise à jour de y : $y^{t+1} = y^t - \rho^t D^t_y$,
  Mise à jour de v : $v^{t+1} = v^t - \rho^t D^t_v$,
  Mise à jour de x : $x^{t+1} = x^t - \gamma^t D^t_x$,
  où D^t_y, D^t_v et D^t_x sont des estimateurs sans biais des directions $D_y(y^t, v^t, x^t)$, $D_v(y^t, v^t, v^t)$ et $D_x(y^t, v^t, x^t)$.
end for

Nous proposons et analysons deux instanciations de cet algorithm général. L'analyse des deux instanciations repose sur les hypothèses de régularité suivantes qui garantissent que la fonction valeur Φ est différentiable avec un gradient lipschitzien et que les directions D_y, D_v et D_x peuvent être controlées.

Hypothèse 6.1. *La fonction f est deux fois différentiable. Les dérivées ∇f et $\nabla^2 f$ sont lipschitziennes en (x, y) avec des constantes de Lipschitz respectives $L_{f,1}$ et $L_{f,2}$.*

Hypothèse 6.2. *La fonction g est trois fois continûment différentiable sur $\mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$. Pour tout $x \in \mathbb{R}^{d_x}$, $g(x, \cdot)$ est μ_g-fortement convexe. Les dérivées ∇g, $\nabla^2 g$ et $\nabla^3 g$ sont lipschitziennes avec des constantes de Lipschitz respectives $L_{g,1}$, $L_{g,2}$ et $L_{g,3}$.*

Hypothèse 6.3. *Il existe $C_f > 0$ tel que pour tout x, nous ayons $\|\nabla_y f(x, y^*(x))\| \leq C_f$.*

La continuité lipschitzienne des dérivées jusqu'au premier ordre pour f et au deuxième ordre pour g est courante dans la littérature (Ji et al., 2021; Arbel and Mairal, 2022a). Cependant, pour éviter de

supposer une taille de batch croissante avec la précision (Arbel and Mairal, 2022a), ou d'éviter de supposer la résolution du système linéaire jusqu'à une précision ϵ dans l'analyse (Chen et al., 2021; Ji et al., 2021), nous supposons la continuité lipschitzienne des dérivées jusqu'au deuxième ordre pour f et au troisième ordre pour g.

Nous faisons également les hypothèses suivantes sur la variance des estimateurs stochastiques.

Hypothèse 6.4. *Il existe B_y, B_v et B_x tels que pour tout t, $\mathbb{E}_t[\|D_y^t\|^2] \leq B_y^2(1 + \|D_y(y^t, v^t, x^t)\|^2)$ et $\mathbb{E}_t[\|D_v^t\|^2] \leq B_v^2(1 + \|D_v(y^t, v^t, x^t)\|^2)$ où $\mathbb{E}_t$ désigne l'espérance conditionnelle à (y^t, v^t, x^t).*

Hypothèse 6.5. *Il existe B_x tel que pour tout t, $\mathbb{E}_t[\|D_x^t\|^2] \leq B_x^2$.*

Algorithme bi-niveaux stochastique (SOBA - *StOchastic Bilevel Algorithm*). Le premier est SOBA, une adaptation de la Descente de Gradient Stochastique (SGD) (Robbins and Monro, 1951) pour les problèmes bi-niveaux. Il consiste à prendre pour les directions

$$\begin{aligned} D_y^t &= \nabla_y g_i(x^t, y^t) \\ D_v^t &= \nabla_{yy}^2 g_i(x^t, y^t) v^t + \nabla_y f_j(x^t, y^t) \\ D_x^t &= \nabla_{xy}^2 g_i(x^t, y^t) v^t + \nabla_x f_j(x^t, y^t) \end{aligned}$$

pour deux indices $i \in [n]$ et $j \in [m]$ tirés aléatoirement.

Pour SOBA, nous fournissons des garanties de convergence avec des tailles de pas fixes qui dépendent de l'horizon (Théorème 6.1). Nous montrons que SOBA atteint un taux de convergence similaire à celui de SGD pour les problèmes non convexes de niveau unique (Ghadimi and Lan, 2013).

Théorème 6.1 (Convergence de SOBA, pas fixes)**.** *Fixons un horizon $T > 1$ et supposons les Hypothèses 6.1 à 6.5 vérfiées. Pour des tailles de pas fixes $\rho^t = \Theta\left(\frac{1}{\sqrt{T}}\right)$ et $\gamma^t = \gamma$ avec $\rho = \mathcal{O}(\sqrt{1/T})$ et $\gamma^t = \Theta\left(\frac{1}{\sqrt{T}}\right)$, les itérés $(x^t)_{t\geq 0}$ de SOBA vérifient*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|\nabla \Phi(x^t)\|^2] = \mathcal{O}(T^{-\frac{1}{2}}) \ .$$

Dans le Théorème 6.2, nous fournissons un taux de convergence pour SOBA avec des tailles de pas décroissantes. Encore une fois, les taux que nous obtenons sont similaires à ceux obtenus pour le SGD dans le cadre non convexe lisse à un seul niveau (Ghadimi and Lan, 2013).

Théorème 6.2 (Convergence de SOBA, pas décroissants)**.** *Supposons que les Hypothèses 6.1 à 4.5 soient vérifiées. Nous considérons des tailles de pas $\rho^t = \Theta\left(\frac{1}{\sqrt{t}}\right)$ et $\gamma^t = \Theta\left(\frac{1}{\sqrt{t}}\right)$. Soit $(x^t)_{t\geq 1}$ la séquence des itérés de SOBA. Alors,*

$$\inf_{t \leq T} \mathbb{E}[\|\nabla \Phi(x^t)\|^2] = \mathcal{O}(\log(T) T^{-\frac{1}{2}}) \ .$$

Algorithme bi-niveaux stochastique moyenné (SABA - *Stochastic Averaged Bilevel Algorithm*). La deuxième instantiation que nous proposons est SABA, une adaptation de l'algorithme réduit de variance SAGA (Defazio et al., 2014) pour les problèmes bi-niveaux. L'idée générale est de remplacer chaque somme dans les directions D par une somme sur une mémoire, en mettant à jour un seul terme à chaque itération. Pour faciliter la présentation, nous notons $z = (y, x, v)$ le vecteur des variables conjointes. Comme nous avons des sommes sur i et sur j, nous avons deux mémoires pour chaque variable : w_i^t pour $i \in [n]$ et $\tilde{w}_j^t$ pour $j \in [m]$, qui gardent la trace des valeurs précédentes de la variable z. À chaque itération t, nous tirons uniformément deux indices indépendants aléatoires $i \in [n]$ et $j \in [m]$ et mettons à jour les mémoires. Pour ce faire, nous posons $w_i^{t+1} = y^t$ et $w_{i'}^{t+1} = w_{i'}^t$ pour $i' \neq i$, et $\tilde{w}_j^{t+1} = y^t$ et $\tilde{w}_{j'}^{t+1} = \tilde{w}_{j'}^t$ pour $j' \neq j$. Chaque somme dans les directions D est alors approximée en utilisant des règles similaires à SAGA : étant donné n fonctions $\phi_{i'}$ pour $i' \in [n]$, nous définissons

$$S[\phi, w]_i^t = \phi_i(w_i^{t+1}) - \phi_i(w_i^t) + \frac{1}{n}\sum_{i'=1}^{n} \phi_{i'}(w_{i'}^t) \ .$$

Ceci est un estimateur sans biais de la moyenne des ϕ puisque $\mathbb{E}_i\left[S[\phi, w]_i^t\right] = \frac{1}{n}\sum_{i=1}^n \phi_i(y^t)$.

Avec un léger abus de notation, nous appelons $\nabla^2_{yy} gv$ la séquence de fonctions $(y \mapsto \nabla^2_{yy} g_i(x,y)v)_{i\in[n]}$ et $\nabla^2_{xy} gv$ la séquence de fonctions $(y \mapsto \nabla^2_{xy} g_i(x,y)v)_{i\in[n]}$. Enfin, les directions stochastiques D_y^t, D_v^t et D_x^t utilisées pour l'algorithme SABA sont données par

$$\begin{aligned} D_y^t &= S[\nabla_y g, w]_i^t \\ D_v^t &= S[\nabla^2_{yy} gv, w]_i^t + S[\nabla_y f, \tilde{w}]_j^t \\ D_x^t &= S[\nabla^2_{xy} gv, w]_i^t + S[\nabla_x f, \tilde{w}]_j^t \quad . \end{aligned}$$

De manière analogue à l'analyse de SAGA, l'analyse de SABA suppose que les hypothèses de régularité s'appliquent à chaque f_j et g_i au lieu de seulement aux sommes f et g.

Hypothèse 6.6. *Pour tout $i \in [n]$ et $j \in [m]$, les fonctions ∇g_i, ∇f_j, $\nabla^2_{yy} g_i$ et $\nabla^2_{xy} g_i$ sont lipschitziennes en (x, y).*

Dans Théorème 6.3, nous montrons que SABA atteint un taux de convergence en $\mathcal{O}((n+m)^{\frac{2}{3}}\epsilon^{-1})$. Ce taux est similaire au taux de convergence de SAGA pour les problèmes non convexes (Reddi et al., 2016) en termes de dépendance au nombre d'échantillons et au nombre d'itérations.

Théorème 6.3 (Convergence de SABA, cas lisse)**.** *Supposons que les Hypothèses 6.1 à 6.3 et l'hypothèse 6.6 soient satisfaites. Nous supposons $\rho = \rho' N^{-\frac{2}{3}}$ et $\gamma = \xi\rho$, où ρ' et ξ dépendent uniquement des constantes de régularité de f et g. Soit $(x^t)_{t\geq 1}$ la suite itératérés de SABA. Alors,*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|\nabla\Phi(x^t)\|^2] = O\left(N^{\frac{2}{3}} T^{-1}\right) \quad .$$

Enfin, dans Théorème 6.4, nous montrons la convergence linéaire de SABA sous une hypothèse de Polyak-Łojasiewicz (PL).

Théorème 6.4 (Convergence de SABA, cas PL)**.** *Supposons que Φ satisfait l'inégalité PL, c'est-à-dire qu'il existe une constante $\mu_\Phi > 0$ telle que pour tout x nous avons $\frac{\mu_\Phi}{2}\|\Phi(x)\|^2 \geq \Phi(x) - \min_x \Phi(x)$. Nous supposons en outre que les Hypothèses 6.1 à 6.3 et 6.5 à 6.6 soient satisfaites. On suppose $\rho = \rho' N^{-\frac{2}{3}}$ et $\gamma = \xi\rho' N^{-1}$, où ρ' et ξ dépendent uniquement de f et g. Soit $(x^t)_{t\geq 0}$ la suite des itérés de SABA et $c' \triangleq \min\left(\mu_\Phi, \frac{1}{16P'}\right)$ avec P' une constant. Alors,*

$$\mathbb{E}[\Phi^T] - \Phi^* = (1 - c'\gamma)^T(\Phi^0 - \Phi^* + C^0)$$

où C^0 est une constante qui dépend de l'initialisation de y, v, x et de la mémoire.

2.2 Une borne inférieure et un algorithme quasi-optimal pour la minimisation du risque empirique à deux niveaux

Nous considérons à nouveau le Problème (6.7) dans le cas où f et g sont des sommes finies telles que dans l'Equation (6.9). Précédemment, nous avons montré dans deux cas spécifiques que des algorithmes bi-niveaux adaptés d'algorithmes à un niveau pouvaient avoir des complexités similaires à leurs homologues à un niveau dans le cadre non convexe/fortement convexe. Cela soulève une question naturelle à laquelle nous essayons de répondre dans (Dagréou et al., 2024b) : Les bornes inférieures et supérieures de complexité se transfèrent-elles du cadre à un niveau au cadre bi-niveaux?

Borne supérieure. Pour les problèmes à un niveau, l'algorithme SARAH (Nguyen et al., 2017) est connu pour être un algorithme quasi-optimal pour la minimisation de sommes finies non convexes lisses (Nguyen et al., 2022). Nous proposons une adaptation de SARAH pour les problèmes bi-niveaux appelée SRBA.

Considérons les dynamiques données par les Équations (6.10), (6.11) et (6.12). L'algorithme SRBA consiste à suivre ces dynamiques avec des estimateurs stochastiques des directions construits de ma-

Algorithm 10 Algorithm stochastic recursif bi-niveaux (SRBA)

Input : initializations $y_0 \in \mathbb{R}^{d_y}$, $x_0 \in \mathbb{R}^{d_x}$, $v_0 \in \mathbb{R}^{d_y}$, number of iterations T and q, step sizes ρ and γ.
Set $\tilde{\mathbf{u}}^{\mathbf{0}} = (y_0, v_0, x_0)$
for $t = 0, \dots, T-1$ **do**
  Reset $\mathbf{\Delta} : \mathbf{\Delta}^{t,0} = (\rho D_y(\tilde{\mathbf{u}}^t), \rho D_v(\tilde{\mathbf{u}}^t), \gamma D_x(\tilde{\mathbf{u}}^t))$
  Update $\mathbf{u} : \mathbf{u}^{t,1} = \Pi(\tilde{\mathbf{u}}^t - \mathbf{\Delta}^{\mathbf{t,0}})$,
  for $k = 1, \dots, q-1$ **do**
    Draw $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$
    $\mathbf{\Delta}_y^{t,k} = \rho(D_{y,i,j}(\mathbf{u}^{t,k}) - D_{y,i,j}(\mathbf{u}^{t,k-1})) + \mathbf{\Delta}_y^{t,k-1}$
    $\mathbf{\Delta}_v^{t,k} = \rho(D_{v,i,j}(\mathbf{u}^{t,k}) - D_{v,i,j}(\mathbf{u}^{t,k-1})) + \mathbf{\Delta}_v^{t,k-1}$
    $\mathbf{\Delta}_x^{t,k} = \gamma(D_{x,i,j}(\mathbf{u}^{t,k}) - D_{x,i,j}(\mathbf{u}^{t,k-1})) + \mathbf{\Delta}_x^{t,k-1}$
    Update $\mathbf{u} : \mathbf{u}^{t,k+1} = \Pi(\mathbf{u}^{t,k} - \mathbf{\Delta}^{t,k})$
  end for
  Set $\tilde{\mathbf{u}}^{t+1} = \mathbf{u}^{t+1,q}$
end for
Return $(\tilde{y}^T, \tilde{v}^T, \tilde{x}^T) = \tilde{\mathbf{u}}^T$

nière récursive et réinitialisés périodiquement avec un calcul complet par lot des directions. L'algorithme est résumé dans l'Algorithme 10 où $\Pi(y, v, x) = (y, \Pi_{B(0,R)}(v), x)$ avec $\Pi_{B(0,R)}$ la projection sur la boule de rayon $R = \sup_{x \in \mathbb{R}^{d_x}} \|v^*(x)\|^2$ centrée en 0. Cette projection permet d'éviter une hypothèse de bornitude sur les itérations v.

L'analyse de convergence de SRBA repose sur les hypothèses de régularité suivantes. Contrairement à l'analyse de SOBA, nos hypothèses de régularité doivent s'appliquer à chaque f_j et g_i au lieu des seules sommes f et g.

Hypothèse 6.7. *Pour tout $j \in [m]$, la fonction f_j est deux fois différentiable et $L_{f,0}$-lipschitzienne. Son gradient est $L_{f,1}$-lipschitzien continu et son Hessian est $L_{f,2}$-Lipschitzien.*

Hypothèse 6.8. *Pour tout $i \in [n]$, la fonction g_i est trois fois différentiable. Ses dérivées de premier, deuxième et troisième ordre sont respectivement $L_{g,1}$-lipschitzienne, $L_{g,2}$-lipschitzienne et $L_{g,3}$-lipschitzienne. Pour $x \in \mathbb{R}^d$, la fonction $g_i(x, \cdot)$ est μ_g-fortement convexe.*

Sur la base de ces hypothèses, nous montrons le taux de convergence de SRBA dans le Théorème 6.5.

Théorème 6.5. *Supposons que les Hypothèses 6.7 et 6.8 soient vérifiées. Supposons que les pas vérifient $\rho \leq \overline{\rho}$ et $\gamma \leq \min(\overline{\gamma}, \xi\rho)$ pour certaines constantes ξ, $\overline{\rho}$ et $\overline{\gamma}$. Alors, on a*

$$\frac{1}{Tq} \sum_{t=0}^{T-1} \sum_{k=0}^{q-1} \mathbb{E}[\|\nabla \Phi(x^{t,k})\|^2] = \mathcal{O}\left(\frac{1}{qT\gamma}\right)$$

où $\mathcal{O}$ cache des constantes de régularité qui sont indépendantes de n et m.

Théorème 6.5 implique que SRBA trouve un point ϵ-stationnaire de Φ après au plus

$$\mathcal{O}\left((n+m)^{\frac{1}{2}} \varepsilon^{-1} \vee (n+m)\right)$$

appels aux oracles, pour un choix approprié des tailles de pas ρ, γ et de la taille de la boucle interne q.

Borne inférieure. Nous établissons également un résultat de borne inférieure sur le nombre d'appels aux oracles nécessaires pour résoudre des problèmes bi-niveaux. Pour cela, il est nécessaire de définir les classes de fonctions et d'algorithmes que nous considérons.

Pour la classe de fonctions, nous considérons les fonctions externes et internes comme étant des sommes finies avec une régularité suffisante pour rendre la fonction valeur Φ différentiable. Cette

classe est formellement définie dans la Définition 6.1.

Définition 6.1. Soient n, m deux entiers positifs, $L_{f,1}$ et μ_g deux constantes positives. La classe des problèmes de minimisation du risque empirique lisse, notée $\mathcal{C}^{L_{f,1},\mu_g}$, est l'ensemble des paires de familles de fonctions à valeurs réelles $((f_j)_{1\leq j\leq m}, (g_i)_{1\leq i\leq n})$ définies sur $\mathbb{R}^{d_y} \times \mathbb{R}^{d_x}$ telles que pour tout $j \in [m]$, f_j est $L_{f,1}$-lisse et pour tout $i \in [n]$, g_i est deux fois différentiable et μ_g-fortement convexe.

La classe d'algorithmes que nous définissons dans la Définition 6.2 est une classe qui inclut plusieurs algorithmes stochastiques basés sur AID de la littérature.

Définition 6.2. Étant donnés les points initiaux y^0, v^0, x^0, un *algorithme linéaire bi-niveaux* $\mathcal{A}$ est une application mesurable telle que pour toute paire $((f_j)_{1\leq j\leq m}, (g_i)_{1\leq i\leq n}) \in \mathcal{C}^{L_{f,1},\mu_g}$, la sortie de l'algorithme $\mathcal{A}((f_j)_{1\leq j\leq m}, (g_i)_{1\leq i\leq n})$ est une séquence $\{(y^t, v^t, x^t, i_t, j_t)\}_{t\geq 0}$ de points (y^t, v^t, x^t) et de variables aléatoires $i_t \in [n]$ et $j_t \in [m]$ telle que pour tout $t \geq 0$

$$\begin{aligned}
y^{t+1} &\in y^0 + \mathrm{Span}\{\nabla_y g_{i_0}(y^0, x^0), \dots, \nabla_y g_{i_t}(y^t, x^t)\} \\
v^{t+1} &\in v^0 + \mathrm{Span}\{\nabla^2_{yy} g_{i_0}(y^0, x^0)v^0 + \nabla_y f_{j_0}(y^0, x^0), \\
&\qquad \dots, \nabla^2_{yy} g_{i_t}(y^t, x^t)v^t + \nabla_y f_{j_t}(y^t, x^t)\} \\
x^{t+1} &\in x^0 + \mathrm{Span}\{\nabla^2_{xy} g_{i_0}(y^0, x^0)v^0 + \nabla_x f_{j_0}(y^0, x^0), \\
&\qquad \dots, \nabla^2_{xy} g_{i_t}(y^t, x^t)v^t + \nabla_x f_{j_t}(y^t, x^t)\}.
\end{aligned}$$

Nous pouvons en déduire une borne inférieure de complexité pour les problèmes bi-niveaux avec ces ingrédients. Nous énonçons le résultat dans le Théorème 6.6.

Théorème 6.6. *Pour tout algorithme linéaire bi-niveau $\mathcal{A}$, et pour tout L^F, n, Δ, ε, p tels que $\varepsilon \leq (\Delta L^F m^{-1})/10^3$, il existe une dimension $d = \mathcal{O}(\Delta\varepsilon^{-1}m^{\frac{1}{2}}L^F)$, un élément $((f_j)_{1\leq j\leq m}, (g_i)_{1\leq i\leq n}) \in \mathcal{C}^{L_{f,1},\mu_g}$ tel que la fonction valeur h définie comme dans (5.1) satisfait $\Phi(x^0) - \inf_{x\in\mathbb{R}^d}\Phi(x) \leq \Delta$ et pour trouver $\hat{x} \in \mathbb{R}^d$ tel que $\mathbb{E}[\|\nabla\Phi(\hat{x})\|^2] \leq \varepsilon$, $\mathcal{A}$ nécessite au moins $\Omega(m^{\frac{1}{2}}\varepsilon^{-1})$ appels aux oracles de la forme (5.10).*

En toute rigueur, les classes de fonctions considérées pour les bornes supérieure et inférieure sont différentes et ne correspondent pas. En effet, pour l'analyse de SRBA, nous devons supposer un peu plus de régularité sur les fonctions f_j et g_i pour contrôler la dynamique de la variable v. De plus, le résultat de la borne inférieure est un résultat partiel puisque toute dépendance dans le nombre de fonctions internes n apparaît.

2.3 Benchmark exhaustif des algorithmes d'optimisation bi-niveaux

La littérature sur l'optimisation bi-niveaux stochastique est riche de nombreuses variantes d'algorithmes accompagnées de garanties théoriques. Cependant, évaluer leur performance pratique est crucial pour comprendre les avantages et les limites de chaque méthode et pour renforcer les résultats théoriques. De nombreux papiers sur l'optimisation bi-niveaux fournissent des résultats empiriques sur des tâches classiques telles que la sélection d'hyperparamètres ou le nettoyage des données. Cependant, le code est souvenet indisponible ou, lorsqu'il est disponible, sa documentation est incomplète, ce qui conduit à un manque de reproductibilité dans le domaine.

Pour ces raisons, nous proposons un benchmark exhaustif des algorithmes d'optimisation bi-niveaux sur plusieurs tâches. Ce benchmark a été réalisé avec le package `Python Benchopt` (Moreau et al., 2022), qui est un outil permettant de construire des benchmarks standardisés et collaboratifs. Le code est ouvert et disponible sur GitHub[3]. De plus, une page HTML affiche les derniers résultats du benchmark[4].

Dans le benchmark, 17 solveurs sont implémentés :

[3] Le code du benchmark est disponible à https://github.com/benchopt/benchmark_bilevel

[4] Les résultats du benchmark sont disponibles à https://benchopt.github.io/results/benchmark_bilevel.html

- **5 solveurs stochastiques sans réduction de variance.** AmIGO (Arbel and Mairal, 2022a), stocBiO (Ji et al., 2021), BSA (Ghadimi and Wang, 2018), TTSA (Hong et al., 2023), SOBA (Dagréou et al., 2022a) ; item **6 solveurs stochastiques avec réduction de variance.** MRBO (Yang et al., 2021), VRBO (Yang et al., 2021), SUSTAIN (Khanduri et al., 2021), FSLA (Li et al., 2022), SABA (Dagréou et al., 2022a), SRBA (Dagréou et al., 2024b) ;
- **3 solveurs sans Hessienne.** BOME (Ye et al., 2022), F2SA (Kwon et al., 2023a), PZOBO (Sow et al., 2022) ;
- **2 solveurs déterministes basés sur Jaxopt (Blondel et al., 2021).** Jaxopt-GD, Jaxopt-ITD ;
- **Un solveur d'ordre zéro.** `Optuna` (Akiba et al., 2019).

Nous considérons trois tâches : un problème jouet avec des fonctions quadratiques, le problème de sélection d'hyperparamètres pour la régression logistique régularisée par régularisation ℓ^2 avec les ensembles de données IJCNN1 et COVTYPE, et la tâche de nettoyage des données avec le jeu de données MNIST.

PUBLICATIONS

International conferences.

As first author:

- Dagréou et al. (2022a): M. Dagréou, P. Ablin, S. Vaiter, and T. Moreau. A framework for bilevel optimization that enables stochastic and global variance reduction algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022;
- Dagréou et al. (2024b): M. Dagréou, T. Moreau, S. Vaiter, and P. Ablin. A Lower Bound and a Near-Optimal Algorithm for Bilevel Empirical Risk Minimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024.

As a middle author:

- Moreau et al. (2022): T. Moreau, M. Massias, A. Gramfort, P. Ablin, P.-A. Bannier, B. Charlier, M. Dagréou, T. D. la Tour, G. Durif, C. F. Dantas, Q. Klopfenstein, J. Larsson, E. Lai, T. Lefort, B. Malézieux, B. Moufad, B. T. Nguyen, A. Rakotomamonjy, Z. Ramzi, J. Salmon, and S. Vaiter. Benchopt: Reproducible, efficient and collaborative optimization benchmarks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

National conferences.

- Dagréou et al. (2022b): M. Dagréou, P. Ablin, S. Vaiter, and T. Moreau. Algorithmes stochastiques et réduction de variance grâce à un nouveau cadre pour l'optimisation bi-niveaux. In *XXVIIIème Colloque Francophone de Traitement Du Signal et Des Images GRETSI*, 2022
- Dagréou et al. (2023): M. Dagréou, T. Moreau, S. Vaiter, and P. Ablin. Borne inférieure de complexité et algorithme quasi-optimal pour la minimisation de risque empirique bi-niveaux. In *XXIXème Colloque Francophone de Traitement Du Signal et Des Images GRETSI*, 2023

Miscellaneous.

We also wrote the following blogpost on the computation of Hessian-vector products. It was published in the blogpost track of ICLR 2024:

- Dagréou et al. (2024a) M. Dagréou, P. Ablin, S. Vaiter, and T. Moreau. How to compute Hessian-vector products?, In *ICLR blogpost track*, 2024, https://iclr-blogposts.github.io/2024/blog/bench-hvp/.

BIBLIOGRAPHY

P. Ablin, G. Peyré, and T. Moreau. Super-efficiency of automatic differentiation for functions defined as a minimum. In *International Conference on Machine Learning (ICML)*, 2020. page 56

A. Agarwal and L. Bottou. A Lower Bound for the Optimization of Finite Sums. In *International Conference on Machine Learning (ICML)*, 2015. page 106

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974. ISSN 0018-9286. page 6

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. In *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2019. pages 7, 58, 150, 157

B. Amos and J. Z. Kolter. OptNet: Differentiable Optimization as a Layer in Neural Networks. In *International Conference on Machine Learning (ICML)*, 2017. page 12

M. Arbel and J. Mairal. Amortized Implicit Differentiation for Stochastic Bilevel Optimization. In *International Conference on Learning Representations (ICLR)*, 2022a. pages 47, 48, 49, 51, 54, 55, 58, 70, 78, 102, 107, 118, 123, 146, 150, 152, 153, 157

M. Arbel and J. Mairal. Non-Convex Bilevel Games with Critical Point Selection Maps. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022b. page 56

F. Bach. Learning theory from the first principles, 2021. page 2

S. Bai, J. Z. Kolter, and V. Koltun. Deep Equilibrium Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. pages 12, 145, 151

J. F. Bard. *Practical Bilevel Optimization: Algorithms and Applications*. Kluwer Acadmic, Dordrecht; London, 2011. page 9

T. Bartz-Beielstein, C. Doerr, D. van den Berg, J. Bossek, S. Chandrasekaran, T. Eftimov, A. Fischbach, P. Kerschke, W. La Cava, M. Lopez-Ibanez, K. M. Malan, J. H. Moore, B. Naujoks, P. Orzechowski, V. Volz, M. Wagner, and T. Weise. Benchmarking in Optimization: Best Practice and Open Issues. *arXiv preprint arXiv:2007.03488*, 2020. page 58

F. L. Bauer. Computational Graphs and Rounding Error. *SIAM Journal on Numerical Analysis*, 11(1): 87–96, 1974. ISSN 0036-1429, 1095-7170. page 30

A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, 18(153):1–43, 2018. page 30

S. Ben-David, N. Eiron, and P. M. Long. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66(3):496–514, 2003. ISSN 00220000. page 2

Y. Bengio. Gradient-Based Optimization of Hyperparameters. *Neural Computation*, 12(8):1889–1900, 2000. ISSN 0899-7667, 1530-888X. page 9

K. Bennett, Jing Hu, Xiaoyun Ji, G. Kunapuli, and Jong-Shi Pang. Model Selection via Bilevel Optimization. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 1922–1929, Vancouver, BC, Canada, 2006. IEEE. pages 6, 9

J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012. page 7

J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox. Hyperopt: A Python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, 2015. page 7

J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011. page 7

C. Bertocchi, E. Chouzenoux, M.-C. Corbineau, J.-C. Pesquet, and M. Prato. Deep unfolding of a proximal interior point method for image restoration. *Inverse Problems*, 36(3):034005, 2020. ISSN 0266-5611, 1361-6420. page 56

Q. Bertrand, Q. Klopfenstein, M. Blondel, S. Vaiter, A. Gramfort, and J. Salmon. Implicit differentiation of Lasso-type models for hyperparameter optimization. In *International Conference on Machine Learning (ICML)*, 2020. page 44

Q. Bertrand, Q. Klopfenstein, M. Massias, M. Blondel, S. Vaiter, A. Gramfort, and J. Salmon. Implicit differentiation for fast hyperparameter selection in non-smooth convex learning. *Journal of Machine Learning Research*, 23(149):1–43, 2022. page 44

M. Blondel, Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-López, F. Pedregosa, and J.-P. Vert. Efficient and Modular Implicit Differentiation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. pages 58, 150, 157

J. Bolte and E. Pauwels. Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning. *Mathematical Programming*, 2020. ISSN 0025-5610, 1436-4646. page 44

J. Bolte, T. Le, E. Pauwels, and A. Silveti-Falls. Nonsmooth Implicit Differentiation for Machine Learning and Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. page 44

J. Bolte, E. Pauwels, and S. Vaiter. Automatic differentiation of nonsmooth iterative algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. page 57

J. Bolte, E. Pauwels, and S. Vaiter. One-step differentiation of iterative algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. page 56

S. Bonettini, L. Calatroni, D. Pezzi, and M. Prato. Algorithmic unfolding for image reconstruction and localization problems in fluorescence microscopy. *arXiv preprint arXiv:2403.17506*, 2024. page 56

L. Bottou. Large-Scale Machine Learning with Stochastic Gradient Descent. In *International Conference on Computational Statistics (COMPSTAT)*, pages 177–186, Heidelberg, 2010. pages 26, 66

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *Siam Reviews*, 60(2):223–311, 2018. page 26

J. Bracken and J. T. McGill. Mathematical Programs with Optimization Problems in the Constraints. *Operations Research*, 21(1):37–44, 1973. ISSN 0030-364X, 1526-5463. page 9

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: Composable transformations of Python+NumPy programs, 2018. pages 33, 58

C. Brauer, N. Breustedt, T. de Wolff, and D. A. Lorenz. Learning Variational Models with Unrolling and Bilevel Optimization. *arXiv preprint arXiv:2209.12651*, 2023. page 56

C. G. Broyden. A class of methods for solving nonlinear simultaneous equations. *Mathematics of computation*, 19(92):577–593, 1965. page 47

S. Bubeck. Convex Optimization: Algorithms and Complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015. ISSN 1935-8237, 1935-8245. page 8

L. Calatroni, C. Cao, J. Carlos De Los Reyes, C.-B. Schönlieb, and T. Valkonen. 8. Bilevel approaches for learning of variational imaging models. In M. Bergounioux, G. Peyré, C. Schnörr, J.-B. Caillau, and T. Haberkorn, editors, *Variational Methods*, pages 252–290. De Gruyter, 2016. page 6

W. Candler and R. Norton. *Multi-Level Programming and Development Policy*. Number vol. 1 in Multi-Level Programming and Development Policy. World Bank, 1977. pages 8, 9

Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford. Lower bounds for finding stationary points I. *Mathematical Programming*, 184(1-2):71–120, 2020. ISSN 0025-5610, 1436-4646. pages 106, 118

Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford. Lower bounds for finding stationary points II: First-order methods. *Mathematical Programming*, 185(1-2):315–355, 2021. ISSN 0025-5610, 1436-4646. pages 118, 119, 120

A. Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées. In *Comptes Rendus de l'Académie Des Sciences de Paris*, 1847. page 23

L. Chen, Y. Ma, and J. Zhang. Near-Optimal Fully First-Order Algorithms for Finding Stationary Points in Bilevel Optimization. *arXiv preprint arXiv:2306.14853*, 2023a. page 57

T. Chen, Y. Sun, and W. Yin. Closing the Gap: Tighter Analysis of Alternating Stochastic Gradient Methods for Bilevel Problems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. pages 50, 51, 54, 55, 70, 72, 73, 74, 82, 85, 128, 146, 153

T. Chen, X. Chen, W. Chen, H. Heaton, J. Liu, Z. Wang, and W. Yin. Learning to optimize: A primer and a benchmark. *Journal of Machine Learning Research*, 23(189):1–59, 2022a. page 55

T. Chen, Y. Sun, and W. Yin. A Single-Timescale Stochastic Bilevel Optimization Method. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2022b. page 51

Z. Chen, B. Kailkhura, and Y. Zhou. An accelerated proximal algorithm for regularized nonconvex and nonsmooth bi-level optimization. *Machine Learning*, 112(5):1433–1463, 2023b. ISSN 0885-6125, 1573-0565. page 47

E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. AutoAugment: Learning Augmentation Strategies from Data. In *arXiv Preprint arXiv:1805.09501*, 2019. pages 145, 151

A. Cutkosky and F. Orabona. Momentum-based variance reduction in non-convex SGD. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. pages 51, 66, 68

M. Dagréou, P. Ablin, S. Vaiter, and T. Moreau. A framework for bilevel optimization that enables stochastic and global variance reduction algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022a. pages 107, 117, 118, 123, 146, 150, 152, 157, 159

M. Dagréou, P. Ablin, S. Vaiter, and T. Moreau. Algorithmes stochastiques et réduction de variance grâce à un nouveau cadre pour l'optimisation bi-niveaux. In *XXVIIIème Colloque Francophone de Traitement Du Signal et Des Images GRETSI*, 2022b. page 159

M. Dagréou, T. Moreau, S. Vaiter, and P. Ablin. Borne inférieure de compléxité et algorithme quasi-optimal pour la minimisation de risque empirique bi-niveaux. In *XXIXème Colloque Francophone de Traitement Du Signal et Des Images GRETSI*, 2023. page 159

M. Dagréou, P. Ablin, S. Vaiter, and T. Moreau. How to compute Hessian-vector products? In *ICLR Blogposts*, 2024a. page 159

M. Dagréou, T. Moreau, S. Vaiter, and P. Ablin. A Lower Bound and a Near-Optimal Algorithm for Bilevel Empirical Risk Minimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024b. pages 148, 150, 154, 157, 159

J. M. Danskin. *The Theory of Max-Min and Its Application to Weapons Allocation Problems*, volume 5 of *Ökonometrie Und Unternehmensforschung / Econometrics and Operations Research*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1967. page 57

A. Defazio and K. Mishchenko. Learning-Rate-Free Learning by D-Adaptation. In *International Conference on Machine Learning (ICML)*, 2023. page 144

A. Defazio, F. Bach, and S. Lacoste-Julien. SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. pages 14, 27, 28, 66, 69, 106, 147, 153

S. Dempe. Directional differentiability of optimal solutions under Slater's condition. *Mathematical Programming*, 59(1-3):49–69, 1993. ISSN 0025-5610, 1436-4646. page 42

S. Dempe. An Implicit Function Approach to Bilevel Programming Problems. In P. M. Pardalos, R. Horst, A. Migdalas, P. M. Pardalos, and P. Värbrand, editors, *Multilevel Optimization: Algorithms and Applications*, volume 20, pages 273–294. Springer US, Boston, MA, 1998. page 42

S. Dempe. *Foundations of Bilevel Programming*. Springer, New York; London, 2011. page 9

S. Dempe, J. Dutta, and B. S. Mordukhovich. New necessary optimality conditions in optimistic bilevel programming. *Optimization*, 56(5-6):577–604, 2007. ISSN 0233-1934, 1029-4945. page 9

S. Dempe, B. Mordukhovich, and A. Zemkoho. Necessary optimality conditions in pessimistic bilevel programming. *Optimization*, 63(4):505–533, 2014. ISSN 0233-1934, 1029-4945. page 9

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2019. page 36

L. Devroye and T. Wagner. Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory*, 25(5):601–604, 1979. ISSN 0018-9448. page 6

J. Domke. Generic methods for optimization-based modeling. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012. page 56

D. L. Donoho and I. M. Johnstone. Adapting to Unknown Smoothness via Wavelet Shrinkage. *Journal of the American Statistical Association*, 90(432):1200–1224, 1995. ISSN 0162-1459, 1537-274X. page 6

A. L. Dontchev and R. T. Rockafellar. *Implicit Functions and Solution Mappings: A View from Variational Analysis*. Springer Monographs in Mathematics. Springer New York, New York, NY, 2009. page 42

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, Georg Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations (ICLR)*, 2021. page 36

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011. page 26

L. El Ghaoui, F. Gu, B. Travacca, A. Askari, and A. Y. Tsai. Implicit Deep Learning. *SIAM Journal on Mathematics of Data Science*, 3:930–958, 2021. page 12

T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019. page 13

C. Fan, G. Choné-Ducasse, M. Schmidt, and C. Thrampoulidis. BiSLS/SPS: Auto-tune Step Sizes for Stable Bi-level Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a. page 144

S. Fan, M. Pagliardini, and M. Jaggi. DoGE: Domain Reweighting with Generalization Estimation. In *International Conference on Machine Learning (ICML)*, 2024b. page 11

C. Fang, C. J. Li, Z. Lin, and T. Zhang. SPIDER: Near-Optimal Non-Convex Optimization via Stochastic Path Integrated Differential Estimator. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. pages 52, 66, 106, 108, 118

V. Feldman, V. Guruswami, P. Raghavendra, and Y. Wu. Agnostic Learning of Monomials by Halfspaces Is Hard. *SIAM Journal on Computing*, 41(6):1558–1590, 2012. ISSN 0097-5397, 1095-7111. page 2

L. Franceschi, M. Donini, P. Frasconi, and M. Pontil. Forward and Reverse Gradient-Based Hyperparameter Optimization. In *International Conference on Machine Learning (ICML)*, 2017. pages 11, 78

L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil. Bilevel Programming for Hyperparameter Optimization and Meta-Learning. In *International Conference on Machine Learning (ICML)*, 2018. pages 6, 145, 151

G. Garrigos and R. M. Gower. Handbook of Convergence Theorems for (Stochastic) Gradient Methods. *arXiv preprint arXiv:2301.11235*, 2023. page 19

S. Geisser. A predictive approach to the random effect model. *Biometrika*, 61(1):101–107, 1974. ISSN 0006-3444, 1464-3510. page 6

S. Ghadimi and G. Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013. pages 26, 106, 147, 153

S. Ghadimi and M. Wang. Approximation Methods for Bilevel Programming. *arXiv preprint arXiv:1802.02246*, 2018. pages 42, 43, 45, 47, 48, 50, 51, 53, 55, 58, 70, 78, 102, 107, 150, 157

J. C. Gilbert. Automatic differentiation and iterative processes. *Optimization Methods and Software*, 1 (1):13–21, 1992. ISSN 1055-6788, 1029-4937. page 56

D. Grangier, P. Ablin, and A. Hannun. Bilevel Optimization to Learn Training Distributions for Language Modeling under Domain Shift. In *Workshop on Distribution Shifts at NeurIPS*, 2023. pages 11, 144

R. Grazzi, L. Franceschi, M. Pontil, and S. Salzo. On the iteration complexity of hypergradient computation. In *International Conference on Machine Learning (ICML)*, 2020. pages 44, 47, 56

R. Grazzi, M. Pontil, and S. Salzo. Convergence properties of stochastic hypergradients. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021. page 78

R. Grazzi, M. Pontil, and S. Salzo. Bilevel Optimization with a Lower-level Contraction: Optimal Sample Complexity without Warm-start. *Journal of Machine Learning Research*, 24(167):1–37, 2023. pages 48, 54, 55

R. Grazzi, M. Pontil, and S. Salzo. Nonsmooth Implicit Differentiation: Deterministic and Stochastic Convergence Rates. In *International Conference on Machine Learning (ICML)*, 2024. page 44

A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Second Edition*. Society for Industrial and Applied Mathematics, second edition, 2008. page 30

Z. Guo, Q. Hu, L. Zhang, and T. Yang. Randomized Stochastic Variance-Reduced Methods for Multi-Task Stochastic Bilevel Optimization. *arXiv preprint arXiv:2105.02266*, 2021a. page 55

Z. Guo, Y. Xu, W. Yin, R. Jin, and T. Yang. On Stochastic Moving-Average Estimators for Non-Convex Optimization. *arXiv preprint arXiv:2104.14840*, 2021b. page 55

L. Hascoet and M. Araya-Polo. Enabling user-driven Checkpointing strategies in Reverse-mode Automatic Differentiation. *arXiv preprint arXiv:0606042*, 2006. page 56

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York, New York, NY, 2009. page 2

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. page 36

M. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409, 1952. ISSN 0091-0635. page 47

A. E. Hoerl and R. W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1):55–67, 1970. ISSN 0040-1706, 1537-2723. page 5

M. Hong, H.-T. Wai, Z. Wang, and Z. Yang. A Two-Timescale Stochastic Algorithm Framework for Bilevel Optimization: Complexity Analysis and Application to Actor-Critic. *SIAM Journal on Optimization*, 33(1):147–180, 2023. ISSN 1052-6234, 1095-7189. pages 51, 55, 58, 70, 72, 74, 78, 102, 103, 150, 157

Q. Hu, Y. Zhong, and T. Yang. Multi-block Min-max Bilevel Optimization with Applications in Multi-task Deep AUC Maximization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. page 109

Y. Huang, E. Chouzenoux, and J.-C. Pesquet. Unrolled Variational Bayesian Algorithm for Image Blind Deconvolution. *IEEE Transactions on Image Processing*, 32:430–445, 2023. ISSN 1057-7149, 1941-0042. page 56

F. Iutzeler, E. Pauwels, and S. Vaiter. Derivatives of Stochastic Gradient Descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. page 56

K. Ji and Y. Liang. Lower Bounds and Accelerated Algorithms for Bilevel Optimization. *Journal of Machine Learning Research*, 24(22):1–56, 2023. pages 47, 106, 118

K. Ji, J. Yang, and Y. Liang. Bilevel Optimization: Convergence Analysis and Enhanced Design. In *International Conference on Machine Learning (ICML)*, 2021. pages 44, 47, 48, 51, 53, 54, 55, 58, 70, 78, 102, 107, 123, 146, 150, 152, 153, 157

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2013. pages 28, 66

H. Th. Jongen, D. Klatte, and K. Tammer. Implicit functions and sensitivity of stationary points. *Mathematical Programming*, 49(1-3):123–138, 1990. ISSN 0025-5610, 1436-4646. page 42

H. Karimi, J. Nutini, and M. Schmidt. Linear Convergence of Gradient and Proximal-Gradient Methods Under the Polyak-Łojasiewicz Condition. In *European Conference on Machine Learning (ECML)*, 2016. pages 114, 129, 130

A. Khaled, K. Mishchenko, and C. Jin. DoWG Unleashed: An Efficient Universal Parameter-Free Gradient Descent Method. *arXiv preprint arXiv:2301.07733*, 2024. page 144

P. Khanduri, S. Zeng, M. Hong, H.-T. Wai, Z. Wang, and Z. Yang. A Near-Optimal Algorithm for Stochastic Bilevel Optimization via Double-Momentum. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. pages 51, 52, 55, 58, 66, 70, 78, 150, 157

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015. page 26

S. G. Krantz, H. R. Parks, and S. G. Krantz. *Implicit Function Theorem: History, Theory, and Applications*. Modern Birkhäuser Classics. Birkhäuser, New York, 2013. page 42

A. Krogh and J. Hertz. A simple weight decay can improve generalization. In J. Moody, S. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, 1991. page 5

J. Kwon, D. Kwon, S. Wright, and R. Nowak. A Fully First-Order Method for Stochastic Bilevel Optimization. In *International Conference on Machine Leaning (ICML)*, 2023a. pages 57, 58, 107, 123, 150, 157

J. Kwon, D. Kwon, S. Wright, and R. Nowak. On Penalty Methods for Nonconvex Bilevel Optimization and First-Order Stochastic Approximation. In *International Conference on Learning Representations (ICLR)*, 2023b. page 57

J. Kwon, D. Kwon, and H. Lyu. On the Complexity of First-Order Methods in Stochastic Bilevel Optimization. In *International Conference on Machine Learning (ICML)*, 2024. page 57

S. K. Lam, A. Pitrou, and S. Seibert. Numba: A LLVM-based Python JIT compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pages 1–6, Austin Texas, 2015. ACM. pages 79, 80

J. Larsen, L. Hansen, C. Svarer, and M. Ohlsson. Design and regularization of neural networks: The optimal use of a validation set. In *Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop*, pages 62–71, Kyoto, Japan, 1996. IEEE. page 9

J. Li, B. Gu, and H. Huang. A Fully Single Loop Algorithm for Bilevel Optimization without Hessian Inverse. In *Proceedings of the Thirty-sixth AAAI Conference on Artificial Intelligence, AAAI'22*, 2022. pages 54, 55, 58, 66, 68, 70, 78, 107, 118, 150, 157

S. Linnainmaa. The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors. *Master's Thesis (University of Helsinki)*, 1970. page 32

S. Linnainmaa. Taylor expansion of the accumulated rounding error. *BIT*, 16(2):146–160, 1976. ISSN 0006-3835, 1572-9125. page 32

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, 1989. ISSN 0025-5610, 1436-4646. pages 47, 80

H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations (ICLR)*, 2019. pages 13, 14, 145, 151

R. Liu, Y. Liu, W. Yao, S. Zeng, and J. Zhang. Averaged Method of Multipliers for Bi-Level Optimization without Lower-Level Strong Convexity. In *International Conference on Machine Learning (ICML)*, 2023. page 57

N. Loizou, S. Vaswani, I. Laradji, and S. Lacoste-Julien. Stochastic Polyak Step-size for SGD: An Adaptive Learning Rate for Fast Convergence. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021. page 144

J. Lorraine, P. Vicol, and D. Duvenaud. Optimizing Millions of Hyperparameters by Implicit Differentiation. In *International Conference on Artificial Intelligence and Statistics (AISTAT)*, 2020. page 6

Z.-Q. Luo, J.-S. Pang, and D. Ralph. *Mathematical Programs with Equilibrium Constraints*. Cambridge University Press, 1 edition, 1996. page 9

D. Maclaurin, D. Duvenaud, and R. P. Adams. Gradient-based Hyperparameter Optimization through Reversible Learning. In *International Conference on Machine Learning (ICML)*, 2015. page 56

B. Malézieux, T. Moreau, and M. Kowalski. Understanding approximate and unrolled dictionary learning for pattern recovery. In *International Conference on Learning Representations (ICLR)*, 2022. page 57

S. Mehmood and P. Ochs. Automatic Differentiation of Some First-Order Methods in Parametric Optimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019. page 56

K. Mishchenko and A. Defazio. Prodigy: An Expeditiously Adaptive Parameter-Free Learner. In *International Conference on Machine Learning (ICML)*, 2024. page 144

J. Mockus. *The Bayesian Approach to Local Optimization*, volume 37, pages 125–156. Springer Netherlands, Dordrecht, 1989. page 7

T. Moreau, M. Massias, A. Gramfort, P. Ablin, P.-A. Bannier, B. Charlier, M. Dagréou, T. D. la Tour, G. Durif, C. F. Dantas, Q. Klopfenstein, J. Larsson, E. Lai, T. Lefort, B. Malézieux, B. Moufad, B. T. Nguyen, A. Rakotomamonjy, Z. Ramzi, J. Salmon, and S. Vaiter. Benchopt: Reproducible, efficient and collaborative optimization benchmarks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. pages 58, 79, 139, 150, 156, 159

E. Moulines and F. Bach. Non-Asymptotic Analysis of Stochastic Approximation Algorithms for Machine Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011. pages 26, 27

A. S. Nemirovsky and D. B. Yudin. *Problem complexity and method efficiency in optimization*. Wiley-Interscience series in discrete mathematics. Wiley, Chichester ; New York, 1983. page 106

I. E. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Number v. 87 in Applied Optimization. Kluwer Academic Publishers, Boston, 2004. page 72

Y. Nesterov. A method for solving the convex programming problem with convergence rate $\mathcal{O}(1/k^2)$. *Proceedings of the USSR Academy of Sciences*, 269:543–547, 1983. page 47

Y. Nesterov. *Lectures on Convex Optimization*. Springer Berlin Heidelberg, New York, NY, 2018. pages 8, 19, 42, 118, 119

L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč. SARAH: A Novel Method for Machine Learning Problems Using Stochastic Recursive Gradient. In *International Conference on Machine Learning (ICML)*, 2017. pages 52, 106, 108, 110, 115, 148, 154

L. M. Nguyen, M. van Dijk, D. T. Phan, P. H. Nguyen, T.-W. Weng, and J. R. Kalagnanam. Finite-sum smooth optimization with SARAH. *Computational Optimization and Applications*, 82(3):561–593, 2022. ISSN 0926-6003, 1573-2894. pages 106, 108, 115, 148, 154

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, New York, 2nd ed edition, 2006. pages 19, 47

P. Ochs, R. Ranftl, T. Brox, and T. Pock. Bilevel optimization with nonsmooth lower level problems. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 654–665. Springer, 2015. page 57

P. Ochs, R. Ranftl, T. Brox, and T. Pock. Techniques for Gradient-Based Bilevel Optimization with Nonsmooth Lower Level Problems. *Journal of Mathematical Imaging and Vision*, 56(2):175–194, 2016. ISSN 0924-9907, 1573-7683. page 57

B. Pascal, S. Vaiter, N. Pustelnik, and P. Abry. Automated Data-Driven Selection of the Hyperparameters for Total-Variation-Based Texture Segmentation. *Journal of Mathematical Imaging and Vision*, 63 (7):923–952, 2021. ISSN 0924-9907, 1573-7683. page 6

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. page 33

B. A. Pearlmutter. Fast Exact Multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994. ISSN 0899-7667, 1530-888X. pages 33, 68, 110

F. Pedregosa. Hyperparameter optimization with approximate gradient. In *International Conference on Machine Learning (ICML)*, 2016. pages 6, 42, 44, 45, 47, 78, 145, 151

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85): 2825–2830, 2011. page 6

J.-C. Pesquet, A. Benazza-Benyahia, and C. Chaux. A SURE Approach for Digital Signal/Image Deconvolution Problems. *IEEE Transactions on Signal Processing*, 57(12):4616–4632, 2009. ISSN 1053-587X, 1941-0476. page 6

I. Petrulionyte, J. Mairal, and M. Arbel. Functional Bilevel Optimization for Machine Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. page 44

J. Pineau, K. Sinha, G. Fried, R. N. Ke, and H. Larochelle. ICLR Reproducibility Challenge 2019. 2019. page 58

B. T. Polyak. *Introduction to optimization*. Translations series in mathematics and engineering. Optimization Software, Publications Division, New York, 1987. page 144

D. Qin, C. Wang, Q. Wen, W. Chen, L. Sun, and Y. Wang. Personalized Federated DARTS for Electricity Load Forecasting of Individual Buildings. *IEEE Transactions on Smart Grid*, pages 1–1, 2023. ISSN 1949-3061. page 13

A. Rajeswaran, C. Finn, S. Kakade, and S. Levine. Meta-Learning with Implicit Gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. pages 145, 151

Z. Ramzi, F. Mannel, S. Bai, J.-L. Starck, P. Ciuciu, and T. Moreau. SHINE: SHaring the INverse Estimate from the forward pass for bi-level optimization and implicit models. In *International Conference on Learning Representations (ICLR)*, 2022. page 47

S. J. Reddi, S. Sra, B. Poczos, and A. Smola. Fast Incremental Method for Nonconvex Optimization. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, IEEE, pages 1971–1977, 2016. pages 28, 66, 75, 77, 106, 148, 154

M. Ren, W. Zeng, B. Yang, and R. Urtasun. Learning to Reweight Examples for Robust Deep Learning. In *International Conference on Machine Learning (ICML)*, 2018. page 11

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951. pages 26, 66, 106, 145, 147, 152, 153

C. Rommel, T. Moreau, J. Paillard, and A. Gramfort. CADDA: Class-wise Automatic Differentiable Data Augmentation for EEG Signals. In *International Conference on Learning Representations (ICLR)*, 2022. pages 145, 151

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. ISSN 0028-0836, 1476-4687. page 32

M. S. Salehi, S. Mukherjee, L. Roberts, and M. J. Ehrhardt. Dynamic Bilevel Learning with Inexact Line Search. *arXiv preprint arXiv:2308.10098*, 2023. page 144

R. Sambharya, G. Hall, B. Amos, and B. Stellato. Learning to warm-start fixed-point optimization algorithms. *Journal of Machine Learning Research*, 25(166):1–46, 2024. page 144

C. Santambrogio, M. Pragliola, A. Lanza, M. Donatelli, and L. Calatroni. Whiteness-based bilevel learning of regularization parameters in imaging. In *European Signal Processing Conference (EUSIPCO)*, 2024. page 6

F. Schaipp, R. Ohana, M. Eickenberg, A. Defazio, and R. M. Gower. MoMo: Momentum Models for Adaptive Learning Rates. In *International Conference on Machine Learning (ICML)*, 2024. page 144

M. Schmidt, N. Le Roux, and F. Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017. ISSN 0025-5610, 1436-4646. pages 28, 66

G. Schwarz. Estimating the Dimension of a Model. *The Annals of Statistics*, 6(2), 1978. ISSN 0090-5364. page 6

D. Scieur, Q. Bertrand, G. Gidel, and F. Pedregosa. The Curse of Unrolling: Rate of Differentiating Through Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. page 56

D. Sculley, J. Snoek, A. Rahimi, and A. Wiltschko. Winner's Curse? On Pace, Progress, and Empirical Rigor. In *ICLR Workshop Track*, 2018. page 57

A. Shaban, C.-A. Cheng, N. Hatch, and B. Boots. Truncated Back-propagation for Bilevel Optimization. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019. page 56

S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA, 2014. page 2

J. Shu, Q. Xie, L. Yi, Q. Zhao, S. Zhou, Z. Xu, and D. Meng. Meta-Weight-Net: Learning an Explicit Mapping For Sample Weighting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. page 11

D. Sow, K. Ji, Z. Guan, and Y. Liang. A Primal-Dual Approach to Bilevel Optimization with Multiple Inner Minima. *arXiv preprint arXiv:2203.01123*, 2022. pages 58, 150, 157

C. M. Stein. Estimation of the Mean of a Multivariate Normal Distribution. *The Annals of Statistics*, 9 (6), 1981. ISSN 0090-5364. page 6

M. Stone. Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 36(2):111–133, 1974. ISSN 1369-7412, 1467-9868. page 6

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. ISSN 00359246. pages 5, 44

A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-posed Problems*. W.H. Winston, 1977. page 5

P. Vicol, J. Lorraine, F. Pedregosa, D. Duvenaud, and R. Grosse. On Implicit Bias in Overparameterized Bilevel Optimization. In *International Conference on Machine Learning (ICML)*, 2022. page 48

H. von Stackelberg. *Marktform Und Gleichgewicht*. Die Handelsblatt-Bibliothek "Klassiker Der Nationalökonomie". J. Springer, 1934. page 9

H. von Stackelberg. *Theory of the Market Economy*. Oxford University Press, 1952. page 9

X. Wang, H. Pham, P. Michel, A. Anastasopoulos, J. Carbonell, and G. Neubig. Optimizing Data Usage via Differentiable Rewards. In *International Conference on Machine Learning (ICML)*, 2019. page 11

R. Wengert. A simple automatic derivative evaluation program. *Communications of the ACM*, 7(8): 463–464, 1964. page 31

W. Wiesemann, A. Tsoukalas, P.-M. Kleniati, and B. Rustem. Pessimistic Bilevel Optimization. *SIAM Journal on Optimization*, 23(1):353–380, 2013. ISSN 1052-6234, 1095-7189. page 9

B. E. Woodworth and N. Srebro. Tight Complexity Bounds for Optimizing Composite Objectives. In *Advances in Neural Information Systems Processing (NeurIPS)*, 2016. page 106

Xiao-Ping Zhang and M. Desai. Adaptive denoising based on SURE risk. *IEEE Signal Processing Letters*, 5(10):265–267, 1998. ISSN 1070-9908, 1558-2361. page 6

J. Yang, K. Ji, and Y. Liang. Provably Faster Algorithms for Bilevel Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. pages 51, 52, 55, 58, 66, 70, 78, 107, 109, 117, 123, 150, 157

M. Ye, B. Liu, S. Wright, P. Stone, and Q. Liu. BOME! Bilevel Optimization Made Easy: A Simple First-Order Approach. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. pages 58, 150, 157

P. Ye, T. He, B. Li, T. Chen, L. Bai, and W. Ouyang. β-DARTS++: Bi-level Regularization for Proxy-robust Differentiable Architecture Search. *arXiv preprint arXiv:2301.06393*, 2023. page 13

M. Zhang, S. W. Su, S. Pan, X. Chang, E. M. Abbasnejad, and R. Haffari. iDARTS: Differentiable Architecture Search with Stochastic Implicit Gradients. In *International Conference on Machine Learning (ICML)*, 2021. page 13

D. Zhou and Q. Gu. Lower Bounds for Smooth Nonconvex Finite-Sum Optimization. In *International Conference on Machine Learning (ICML)*, 2019. pages 106, 118, 120

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Contributions à l'optimisation bi-niveaux stochastique

Mots clés : Optimisation bi-niveaux, differenciation implicite, algorithmes stochastiques, apprentissage automatique

Résumé : Les problèmes bi-niveaux sont un type de problèmes d'optimisation caractérisés par une structure hiérarchique. Dans ces problèmes, on cherche à minimiser une fonction externe sous la contrainte que certaines variables minimisent une fonction interne. Ces problèmes gagnent en popularité dans la communauté du machine learning en raison de leur large éventail d'applications, telles que l'optimisation d'hyperparamètres. Dans cette thèse, nous explorons des algorithmes basés sur la différenciation implicite approximée pour aborder les problèmes bi-niveaux où les fonctions externes et internes sont des moyennes empiriques sur des ensembles d'échantillons potentiellement vastes. Ce cadre de minimisation du risque empirique est une approche classique dans de nombreuses tâches de d'apprentissage statistique. Tout d'abord, nous introduisons un cadre algorithmique général qui permet d'adapter n'importe quel solveur stochastique de premier ordre, initialement conçu pour des problèmes à un seul niveau, aux problèmes bi-niveaux. Nous fournissons et analysons deux instanciations de ce cadre : une adaptation de la descente de gradient stochastique et de l'algorithme SAGA aux problèmes bi-niveaux. Notre analyse démontre que ces algorithmes ont des complexités comparables à celles de leurs homologues à un seul niveau. Ensuite, nous nous intéressons à la complexité de l'optimisation bi-niveaux dans le cadre non convexe/fortement convexe. Nous proposons une classe d'algorithmes qui inclut plusieurs méthodes stochastiques basées sur la différenciation implicite approximée et établissons une borne inférieure sur le nombre d'appels aux oracles nécessaires pour atteindre un point stationnaire approché. Nous proposons ensuite un algorithme dont la complexité correspond à cette borne inférieure. Les performances des méthodes proposées sont évaluées numériquement par un benchmark comparant ces méthodes à d'autres algorithmes bi-niveaux sur des fonctions quadratiques, des problèmes de sélection d'hyperparamètres et la tâche de nettoyage de données.

Title: Contributions to stochastic bilevel optimization

Keywords: Bilevel optimization, implicit differentiation, stochastic algorithms, machine learning

Abstract: Bilevel problems are a type of optimization problem characterized by a hierarchical structure. In these problems, one wants to minimize an outer function under the constraint that some variables minimize an inner function. These problems are gaining popularity in the machine learning community due to their wide range of applications, such as hyperparameter optimization. In this thesis, we explore approximate implicit differentiation-based algorithms to address bilevel problems where both the outer and inner functions are empirical means over potentially large sample sets. This empirical risk minimization framework is a classical approach in many machine learning tasks. First, we introduce a general algorithmic framework that allows any first-order stochastic solver, initially designed for single-level problems, to be adapted to bilevel problems. We provide and analyze two instantiations of this framework: an adaptation of the stochastic gradient descent and of the SAGA algorithm to bilevel problems. Our analysis demonstrates that these algorithms have complexities comparable to their single-level counterparts. Then, we interest ourselves in the complexity of bilevel optimization in the nonconvex/strongly convex setting. We propose an algorithm class that includes several stochastic approximate implicit differentiation-based methods and establish a lower bound on the number of oracle calls required to reach an approximate stationary point. Then, we propose an algorithm whose complexity matches this lower bound. The performances of the proposed methods are evaluated numerically by a benchmark comparing those methods to other bilevel algorithms on quadratic functions, hyperparameter selection problems, and the datacleaning task.