Borne inférieure de compléxité et algorithme quasi-optimal pour la minimisation de risque empirique bi-niveaux

Mathieu DAGRÉOU[1] , Thomas MOREAU[1] , Samuel VAITER[2] , Pierre ABLIN[3]

[1]Inria Saclay - 1 Rue Honoré d'Estienne d'Orves, 91120 Palaiseau, France

[2]Laboratoire J.A. Dieudonné, CNRS - Parc Valrose, 06108 Nice Cedex 2, France

[3]Apple

Résumé – L'optimisation à deux niveaux est un thème animant de plus en plus les communautés d'apprentissage statistique et de traitement du signal. Dans de nombreuses applications en apprentissage, les fonctions internes et externes sont des moyennes empiriques. Avec une grande quantité de données, les méthodes stochastiques sont des méthodes de choix pour la minimisation de risque empirique. Nous proposons ici une borne inférieure sur le nombre d'appels aux oracles nécessaire pour résoudre ce problème avec une précision ϵ. Aussi, nous donnons un algorithme dont la complexité atteint cette borne inférieure. En ce sens, cet algorithme est quasi-optimal.

Abstract – Bilevel optimization is increasingly drawing the attention of the machine learning and signal processing communities. In many learning applications, the inner and outer functions are empirical means. In a large sample size setting, stochastic methods are cornerstones for empirical risk minimization. In this paper, we provide a lower bound on the minimum number of calls to oracles needed to solve this problem with an accuracy ϵ. Moreover, we proposed an algorithm whose complexity matches this lower bound. As a consequence, this algorithm is near-optimal.

1 Introduction

Les problèmes d'optimisation bi-niveaux sont des problèmes d'optimisation dans lesquels deux problèmes d'optimisation sont imbriqués. Ces problèmes rencontrent de nombreuses applications en apprentissage automatique. Parmi elles, on peut citer la sélection d'hyperparamètres [13], le méta-apprentissage [4] ou encore l'augmentation de données [14].

Étant données deux fonctions F et G définies sur $\mathbb{R}^p \times \mathbb{R}^d$, on cherche à minimiser par rapport à $z \in \mathbb{R}^p$ et $x \in \mathbb{R}^d$ la quantité $F(z, x)$ sous la contrainte que z est un minimiseur de $G(\bullet, x)$. Lorsque G est fortement convexe par rapport à z, ce problème peut s'écrire sous la forme

$$\begin{aligned} &\min_{x \in \mathbb{R}^d} h(x) = F(z^*(x), x) \ , \\ &\text{avec } z^*(x) \in \operatorname*{arg\,min}_{z \in \mathbb{R}^p} G(z, x) \ . \end{aligned} \tag{1}$$

Par exemple, pour la sélection d'hyperparamètres d'un modèle d'apprentissage supervisé, z est le paramètre du modèle que l'on souhaite apprendre et x est l'hyperparamètre. La fonction G est la fonction de perte d'entraînement qui évalue l'erreur d'apprentissage sur les échantillons d'entraînement. La fonction F est la fonction de perte de validation qui estime l'erreur de généralisation via les échantillons de validation.

Dans le cas où G est fortement convexe par rapport à z et deux fois différentiable et F est différentiable, le théorème des fonctions implicites fournit une expression du gradient de h :

$$\nabla h(x) = \nabla_2 F(z^*(x), x) + \nabla^2_{21} G(z^*(x), x) v^*(x) \ , \tag{2}$$

le vecteur $v^*(x) \in \mathbb{R}^p$ étant défini par

$$v^*(x) = -[\nabla^2_{11} G(z^*(x), x)]^{-1} \nabla_1 F(z^*(x), x) \ . \tag{3}$$

Dans le cas hypothétique où l'accès à $z^*(x)$ et $v^*(x)$ est peu coûteux, le Problème (1) est en fait un problème de minimisation d'une fonction lisse non-convexe. Néanmoins en pratique, on préfère remplacer $z^*(x)$ et $v^*(x)$ par des approximations obtenues après une ou plusieurs itérations de solveurs résolvant les problèmes dont $z^*(x)$ et $v^*(x)$ sont la solution. C'est ce que l'on appelle la différentiation implicite approximée.

Dans de nombreux cas d'usage tels que la sélection d'hyperparamètres, les fonctions F et G peuvent s'écrire comme des moyennes empiriques. Autrement dit, il existe deux entiers m et n tels que pour tout $(z, x) \in \mathbb{R}^p \times \mathbb{R}^d$ on ait

$$F(z, x) = \frac{1}{m} \sum_{j=1}^{m} F_j(z, x), \quad G(z, x) = \frac{1}{n} \sum_{i=1}^{n} G_i(z, x) \ .$$

Dans ce cas, les méthodes stochastiques sont les plus prometteuses pour résoudre le Problème (1).

On peut donc se demander quel est nombre minimal d'opérations nécessaires pour approcher une solution du Problème (1) avec une précision ϵ. Dans le cadre de l'optimisation simple niveau classique, il est établi le nombre minimal d'appels aux oracles (valeur de la fonction et gradient) pour obtenir une solution avec une précision ϵ est $\Omega(\sqrt{m}\epsilon^{-1})$ où m est le nombre de fonctions impliquées dans la moyenne empirique [16, 3]. Cette borne est par ailleurs atteinte par les algorithmes SPIDER [3] et SARAH [10, 11], rendant de fait ces algorithmes optimaux dans ce contexte. On peut alors se demander si ces résultats peuvent être étendus au cas bi-niveaux. Nous répondons par l'affirmative en proposant en Section 2 l'algorithme SRBA pouvant résoudre le Problème (1) en effectuant $\mathcal{O}(\sqrt{n+m}\epsilon^{-1} \vee (n+m))$ appels aux oracles pour trouver un point $\hat{x}$ vérifiant $\mathbb{E}[\|\nabla h(\hat{x})\|^2] \leq \epsilon$, où l'espérance porte sur l'aléa introduit par l'algorithme. Nous montrons ensuite en Section 3 que pour tout algorithme bi-niveaux basé sur la

différentiation implicite approximée il existe deux fonctions F et G telles que l'algorithme trouve un point ϵ-stationnaire de la fonction h en au moins $\Omega(\sqrt{m}\epsilon^{-1})$ appels aux oracles. Ainsi, l'algorithme que nous proposons est quasi-optimal dans le régime $m \approx n$. Enfin, en Section 4, nous illustrons les performances en pratique de SRBA sur un problème de sélection du paramètre de régularisation dans un problème de régression logistique avec une pénalisation ℓ^2.

2 SRBA : Algorithme pour la minimisation de risque empirique bi-niveaux

2.1 Hypothèses

Nous commençons par formuler des hypothèses sur la régularité des fonctions F et G qui sont nécessaires pour la validité de l'algorithme SRBA. Celles-ci sont analogues à celles que l'on trouve dans [2].

Hypothèse 1. Pour tout $j \in [m]$, la fonction F_j est deux fois dérivable, lipschitzienne et ses dérivées ∇F_j et $\nabla^2 F_j$ sont lipschitziennes.

Hypothèse 2. Pour tout $i \in [n]$, la fonction G_i est trois fois dérivable et ses dérivées ∇G, $\nabla^2 G$ et $\nabla^3 G$ sont lipschitziennes. Il existe $\mu_G > 0$ tel que pour tout $x \in \mathbb{R}^d$, la fonction $G(\bullet, x)$ est μ_G-fortement convexe.

La forte convexité de G assure le caractère bien posé du problème (1) et la validité du gradient de h (2). Combinée au caractère lipschitzien de F, elle implique la bornitude de l'application v^* qui est un élément essentiel de notre algorithme.

Proposition 1. *Sous les hypothèses* (1) *et* (2)*, il existe* $R > 0$ *tel que pour tout* $x \in \mathbb{R}^d$, $\|v^*(x)\| \leq R$.

Dorénavant, pour $(z, v, x) \in \mathbb{R}^p \times \mathbb{R}^p \times \mathbb{R}^d$, on note $\Pi(z, v, x) \triangleq (z, v', x)$ où v' est la projection de v sur $B(0, R)$.

2.2 Différentiation implicite approximée

Lorsque F et G vérifient les hypothèses 1 et 2, la fonction h définie dans (1) est différentiable à gradients lipschitziens. Dans les problèmes d'optimisation classiques, une méthode de choix est la descente de gradient. Néanmoins, le gradient de h donné par (2) est coûteux à calculer car il nécessite le calcul de $z^*(x)$ et $v^*(x)$. Une solution courante consiste à remplacer $z^*(x)$ et $v^*(x)$ par des approximations [1, 6]. Cela motive l'introduction de la direction

$$D_x(z, v, x) = \nabla^2_{21} G(z, x)v + \nabla_2 F(z, x) \ .$$

Cette direction vérifie $D_x(z^*(x), v^*(x), x) = \nabla h(x)$ pour tout $x \in \mathbb{R}^d$. Pour approximer z^* et v^*, il est naturel de les faire se déplacer en direction de leurs équillibres respectifs, motivant ainsi l'introduction des directions D_z et D_v

$$\begin{aligned} D_z(z, v, x) &= \nabla_1 G(z, x) \\ D_v(z, v, x) &= \nabla^2_{11} G(z, x)v + \nabla_1 F(z, x) \ . \end{aligned}$$

Ces directions sont linéaires par rapport à F et G. On peut donc en construire des estimateurs stochastiques en considérant pour $i \in [n]$ et $j \in [m]$ les directions échantillonnées

$$\begin{aligned} D_{z,i,j}(z, v, x) &= \nabla_1 G_i(z, x) \\ D_{v,i,j}(z, v, x) &= \nabla^2_{11} G_i(z, x)v + \nabla_1 F_j(z, x) \\ D_{x,i,j}(z, v, x) &= \nabla^2_{21} G_i(z, x)v + \nabla_2 F_j(z, x) \ . \end{aligned}$$

2.3 L'algorithme SRBA

L'algorithme SRBA (*Stochastic Recursive Bilevel Algorithm*) que nous proposons est une adaptation de l'algorithme SARAH [10] pour les problèmes bi-niveaux. Il repose sur une estimation stochastique récursive des directions D_z, D_v, D_x, avec une réinitialisation périodique des estimateurs via un calcul exact des directions.

Dans la suite, on note $\mathbf{u}$ la variable jointe $\mathbf{u} = (z, v, x)$ et $\mathbf{\Delta} = (\rho D_z, \rho D_v, \gamma D_x)$ l'estimée de direction pondéré par la taille du pas. À l'itération t, l'estimée de direction $\mathbf{\Delta}$ est initialisée par le calcul des directions exactes, *i.e.* $\mathbf{\Delta}^{t,0} = (\rho D_z(\tilde{\mathbf{u}}^t), \rho D_v(\tilde{\mathbf{u}}^t), \rho D_x(\tilde{\mathbf{u}}^t))$. Une mise à jour de la variable $\tilde{\mathbf{u}}^t$ est effectuée en se déplaçant dans la direction $-\mathbf{\Delta}^{t,0}$. Puis la variable v est projetée sur la boule fermée $B(0, R)$. On entre ensuite dans une boucle intérieure de taille $q - 1$. À la k^{e} itération de cette boucle, deux indices $i \in \{1, \dots, n\}$ et $j \in \{1, \dots, m\}$ sont tirés aléatoirement et les composantes de l'estimée de direction $\mathbf{\Delta}^{t,k-1} = (\mathbf{\Delta}^{t,k-1}_z, \mathbf{\Delta}^{t,k-1}_v, \mathbf{\Delta}^{t,k-1}_x)$ sont mises à jour selon les équations (4), (5) et (6).

$$\mathbf{\Delta}^{t,k}_z = \rho(D_{z,i,j}(\mathbf{u}^{t,k}) - D_{z,i,j}(\mathbf{u}^{t,k-1})) + \mathbf{\Delta}^{t,k-1}_z \quad (4)$$

$$\mathbf{\Delta}^{t,k}_v = \rho(D_{v,i,j}(\mathbf{u}^{t,k}) - D_{v,i,j}(\mathbf{u}^{t,k-1})) + \mathbf{\Delta}^{t,k-1}_v \quad (5)$$

$$\mathbf{\Delta}^{t,k}_x = \gamma(D_{x,i,j}(\mathbf{u}^{t,k}) - D_{x,i,j}(\mathbf{u}^{t,k-1})) + \mathbf{\Delta}^{t,k-1}_x \ . \quad (6)$$

Puis, $\mathbf{u}$ est mise à jour via la formule $\mathbf{u}^{t,k+1} = \Pi(\mathbf{u}^{t,k} - \mathbf{\Delta}^{t,k})$. L'algorithme est résumé dans l'Algorithme 1. L'utilisation de la projection sur v à chaque itération a été introduite dans [5]. Elle permet d'assurer la bornitude des itérés $(v^{t,k})_{t,k}$. Sans cela, l'analyse théorique de SRBA nécessite de la supposer *a priori*, ce qui serait une hypothèse forte.

La réinitialisation périodique des directions par un calcul exact peut sembler prohibitive. Elle est néanmoins essentielle pour maintenir des estimées suffisamment précises durant toute la procédure. Ce type de technique de est d'ailleurs notamment employée dans d'autres algorithmes d'optimisation stochastiques célèbres tels que SVRG [7] et SARAH [10].

2.4 Complexité de SRBA

Les problèmes bi-niveaux étant non-convexes, on ne peut qu'assurer la convergence vers un point stationnaire de la fonction h. Ci-après, on rappelle la notion de point ϵ-stationnaire.

Définition 1. Soit h une fonction différentiable définie sur $\mathbb{R}^d$ et $\hat{x}$ une variable aléatoire à valeur dans $\mathbb{R}^d$. On dit que $\hat{x}$ est un point ϵ-stationnaire de h si on a $\mathbb{E}[\|\nabla h(\hat{x})\|^2] \leq \epsilon$ où l'espérance est prise par rapport à la loi de $\hat{x}$.

Comme souvent en optimisation différentiable, la performance de notre algorithme est donnée en nombre d'appels aux oracles pour obtenir un point ϵ-stationnaire. En bi-niveaux, un appel à un oracle est le calcul d'une quantité de la forme

$$[\nabla F_j(z, x), \nabla G_i(z, x), \nabla^2_{11} G_i(z, x)v, \nabla^2_{21} G_i(z, x)v].$$

Algorithme 1 : SRBA

Input : $z_0 \in \mathbb{R}^p$, $x_0 \in \mathbb{R}^d$, $v_0 \in \mathbb{R}^p$, $T \in \mathbb{N}^*$, $q \in \mathbb{N}^*$, $\rho > 0$ et $\gamma > 0$.
Initialisation $\tilde{\mathbf{u}}^{\mathbf{0}} = (z_0, v_0, x_0)$
for $t = 0, \dots, T-1$ **do**
  Réinitialisation de $\mathbf{\Delta}$:

$$\mathbf{\Delta}^{t,0} = (\rho D_z(\tilde{\mathbf{u}}^t), \rho D_v(\tilde{\mathbf{u}}^t), \gamma D_x(\tilde{\mathbf{u}}^t))$$

  Mise à jour de $\mathbf{u}$: $\mathbf{u}^{t,1} = \Pi(\tilde{\mathbf{u}}^t - \mathbf{\Delta}^{\mathbf{t},\mathbf{0}})$,
  for $k = 1, \dots, q-1$ **do**
    Tirer $i \in \{1, \dots, n\}$ et $j \in \{1, \dots, m\}$
    Mise à jour de la direction $\mathbf{\Delta}$ via les équations (4), (5) et (6).
    Mise à jour de $\mathbf{u}$: $\mathbf{u}^{t,k+1} = \Pi(\mathbf{u}^{t,k} - \mathbf{\Delta}^{t,k})$
  end for
  Mise à jour $\tilde{\mathbf{u}}^{t+1} = \mathbf{u}^{t+1,q}$
end for
Renvoyer $(\tilde{z}^T, \tilde{v}^T, \tilde{x}^T) = \tilde{\mathbf{u}}^T$

Notons qu'avec la différentiation automatique, les calculs de produits Hessienne-vecteur et Jacobienne-vecteur ont une complexité analogue à celle de l'évaluation du gradient de G [12].

Les Hypothèses 1 et 2 nous permettent d'établir la convergence de SRBA vers un point stationnaire de h.

Théorème 1. *Supposons les hypothèses 1 et 2 vérifiées. Pour des tailles de pas ρ et γ suffisamment petites, les itérés de SRBA vérifient*

$$\frac{1}{Tq}\sum_{t=0}^{T-1}\sum_{k=0}^{q-1}\mathbb{E}[\|\nabla h(x^{t,k})\|^2] = \mathcal{O}\left(\frac{1}{qT\gamma}\right)$$

où $\mathcal{O}$ cache des constantes de régularité indépendantes des nombres d'échantillons m et n.

On voit notamment qu'en prenant $q = 1$ on retrouve un taux de convergence analogue à celui de la descente de gradient pour les problèmes non-convexes. En choisissant la période q et les tailles de pas ρ et γ de manière optimale, on obtient la complexité suivante pour SRBA.

Corollaire 1. *Si les hypothèses du Théorème 1 sont vérifiées et si de plus $\rho = \overline{\rho}(n+m)^{-\frac{1}{2}}$, $\rho = \overline{\gamma}(n+m)^{-\frac{1}{2}}$ pour des constante $\overline{\rho}$ et $\overline{\gamma}$ strictement positives indépendantes de n et de m et $q = n + m$, alors the nombre d'appels aux oracles suffisant pour trouver un point ϵ-stationnaire est d'au plus $\mathcal{O}((n+m)^{\frac{1}{2}}\epsilon^{-1} \vee (n+m))$.*

Dans la section suivante, on montre que cette complexité est quasi-optimale pour les problèmes bi-niveaux.

3 Borne inférieure pour la minimisation de risque empirique bi-niveaux

Nous allons ici donner un résultat de borne inférieure de complexité pour la résolution de problèmes bi-niveaux.

3.1 Classe de problèmes et d'algorithmes

L'établissement d'un résultat de borne inférieure nécessite en premier lieu la définition de la classe de problèmes et d'algorithmes que l'on considère.

Définition 2. Soient $n, m \in \mathbb{N}^*$, $L_1^F, \mu_G > 0$. La classe des problèmes bi-niveaux est l'ensemble $\mathcal{C}^{L_1^F,\mu_G}$ des paires de familles de fonctions $((F_j)_{j\in[m]}, (G_i)_{i\in[n]})$ définies sur $\mathbb{R}^p \times \mathbb{R}^d$ telles que pour tout $j \in [m]$, F_j est différentiable à gradient lipschitzien et pour tout $i \in [n]$, G_i est deux fois differentiable et μ_G-fortement convexe par rapport à sa première variable.

Notons que nous ne faisons aucune hypothèse de convexité sur les F_j. Par conséquent, la classe de problèmes que l'on considère est une classe de problèmes non-convexes.

On considère également les algorithmes basés sur la differentiation implicite approximée.

Définition 3. Étant donnés trois points initiaux z^0, v^0, x^0, un algorithme bi-niveaux linéaire $\mathcal{A}$ est une fonction mesurable telle que pour tout $((F_j)_{j\in[m]}, (G_i)_{i\in[n]}) \in \mathcal{C}^{L_1^F,\mu_G}$, $\mathcal{A}((F_j)_{j\in[m]}, (G_i)_{i\in[n]})$ est une suite $(z^t, v^t, x^t, i_t, j_t)_{t\geq 0}$ de points $(z^t, v^t, x^t) \in \mathbb{R}^{p+p+d}$ et de variables aléatoires (i_t, j_t) à valeurs dans $[n] \times [m]$ tels que pour tout $t \geq 0$ on ait

$$\begin{aligned}
z^{t+1} &\in z^0 + \mathrm{Vect}\{\nabla_1 G_{i_0}(z^0, x^0), \dots, \nabla_1 G_{i_t}(z^t, x^t)\} \\
v^{t+1} &\in v^0 + \mathrm{Vect}\{\nabla^2_{11} G_{i_0}(z^0, x^0)v^0 + \nabla_1 F_{j_0}(z^0, x^0), \\
&\qquad \dots, \nabla^2_{11} G_{i_t}(z^t, x^t)v^t + \nabla_1 F_{j_t}(z^t, x^t)\} \\
x^{t+1} &\in x^0 + \mathrm{Vect}\{\nabla^2_{21} G_{i_0}(z^0, x^0)v^0 + \nabla_2 F_{j_0}(z^0, x^0), \\
&\qquad \dots, \nabla^2_{21} G_{i_t}(z^t, x^t)v^t + \nabla_2 F_{j_t}(z^t, x^t)\}.
\end{aligned}$$

Cette classe d'algorithmes comprend de nombreux algorithmes bi-niveaux tels qu'AmIGO [1], FSLA [8], SOBA [2], SABA [2] et SRBA (Algorithme 1).

3.2 Borne inférieure

Nous pouvons maintenant énoncer le théorème. Il s'agit d'une adaptation du Théorème 4.7 de [16] au cas bi-niveaux.

Théorème 2. *Pour tout algorithme bi-niveaux linéaire $\mathcal{A}$, pour toutes constantes L_1^F, m, Δ, ϵ, p telles que $\epsilon \leq (\Delta L_1^F m^{-1})/10^{-3}$, il existe une dimension $d = \mathcal{O}(\Delta\epsilon^{-1}m^{\frac{1}{2}}L_1^F)$ et un élément $((F_j)_{j\in[m]}, (G_i)_{i\in[n]}) \in \mathcal{C}^{L_1^F,\mu_G}$ tels que la fonction h définie à partir des F_j et G_i comme dans l'Équation (1) vérifie $h(x^0) - \inf_{x\in\mathbb{R}^d} h(x) \leq \Delta$ et que l'algorithme $\mathcal{A}$ nécessite au moins $\Omega(\sqrt{m}\epsilon^{-1})$ appels aux oracles pour trouver un point $\hat{x}$ vérifiant $\mathbb{E}[\|\nabla h(\hat{x})\|^2] \leq \epsilon$.*

La preuve consiste à construire F en prenant $F_j(z, x) = f_j(z)$ où f_j est la fonction "pire cas" utilisée dans [16] pour établir le théorème 4.7. Pour G_i, on prend $G_i(z, x) = \frac{\mu_G}{2}\|z - x\|^2$. Une limitation de notre résultat est l'absence de dépendance en n, le nombre d'échantillons utilisés pour construire G. Ce théorème montre néanmoins que SRBA est un algorithme quasi optimal dans le régime $m \approx n$.

4 Expérience numérique

Afin d'illustrer l'intérêt pratique de SRBA, on propose de l'expérimenter sur un problème de sélection du paramètre de régularisation d'une régression logistique avec régularisation ℓ^2. On considère le *dataset* IJCNN1[1] qui contient des

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/%20datasets/binary.html

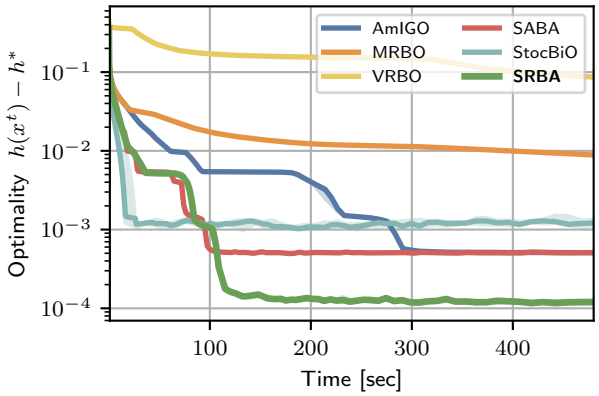

FIGURE 1 : Comparaison de SRBA avec d'autres algorithmes de la littérature d'optimisation bi-niveaux stochastique. Les méthodes sont lancées 10 fois avec 10 graines différentes. Les courbes affichées correspondent alors aux performances médianes. On voit que SRBA atteint le plateau le plus bas parmi les méthodes comparées.

échantillons labelisés dans deux classes. On souhaite entraîner un classifieur binaire sur ces données. Pour cela, on considère les échantillons d'entraînement $(x_i^{\text{train}}, y_i^{\text{train}})_{i\in[n_{\text{train}}]}$ (ici $n_{\text{train}} = 49990$) et la fonction de perte d'entraînement G définie par

$$G(\theta, \lambda) = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \varphi(y_i^{\text{train}} \langle x_i^{\text{train}}, \theta \rangle) + \frac{1}{2} \sum_{k=1}^{p} \mathrm{e}^{\lambda_k} \theta_k^2$$

où $\varphi(u) = \log(1 + \mathrm{e}^{-u})$ et p est le nombre de caractéristiques, ici $p = 22$. Notons qu'ici nous utilisons un paramètre de régularisation différent par variable, ce qui rend trop coûteux une recherche du meilleur λ par grille. Ainsi, pour sélectionner $\lambda \in \mathbb{R}^p$, on dispose d'échantillons de validation $(x_i^{\text{val}}, y_i^{\text{val}})_{j\in[n_{\text{val}}]}$ (ici $n_{\text{val}} = 91701$), ce qui permet de définir la fonction de perte de validation

$$F(\theta, \lambda) = \frac{1}{n_{\text{val}}} \sum_{j=1}^{n_{\text{val}}} \varphi(y_j^{\text{val}} \langle x_j^{\text{val}}, \theta \rangle) \ .$$

Nous résolvons le Problème (1) défini par les fonction F et G à l'aide des algorithmes AmIGO [1], MRBO [15], SABA [2], StocBiO [6], VRBO [15] et SRBA à l'aide du package Python Benchopt [9]. Les différents paramètres de ces algorithmes sont sélectionnés par une recherche par grille, en gardant pour chaque algorithme, le jeu de paramètres qui donne la valeur de h la plus basse parmi tous les itérés.

En Figure 1, on montre l'évolution de la sous-optimalité en fontion du temps pour chacun des algorithmes. On voit que si SRBA n'est pas nécessairement l'algorithme plus rapide sur les premières itérations, il est celui qui atteint la meilleure valeur par rapport à ses concurrents.

5 Conclusion

Nous avons proposé SRBA, un nouvel algorithme pour la minimisation de risque empirique à deux niveaux. Cet algorithme a une complexité qui correspond à la complexité minimale pour résoudre de tels problèmes. Ainsi, SRBA est quasi-optimal. Par ailleurs, la borne inférieure que nous avons établie est analogue à la borne inférieure des problèmes de minimisation de risque empirique simple niveau non-convexes. On peut ainsi dire que les problèmes bi-niveaux considérés dans cet article sont aussi difficiles que les problèmes simples niveaux non-convexes. Enfin, nous avons illustré expérimentalement l'intérêt pratique de SRBA.

Références

[1] Michael ARBEL et Julien MAIRAL : Amortized Implicit Differentiation for Stochastic Bilevel Optimization. *In International Conference on Learning Representations (ICLR)*, 2022.

[2] Mathieu DAGRÉOU, Pierre ABLIN, Samuel VAITER et Thomas MOREAU : A framework for bilevel optimization that enables stochastic and global variance reduction algorithms. *In Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[3] Cong FANG, Chris Junchi LI, Zhouchen LIN et Tong ZHANG : SPIDER : Near-Optimal Non-Convex Optimization via Stochastic Path Integrated Differential Estimator. *In Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[4] Luca FRANCESCHI, Paolo FRASCONI, Saverio SALZO, Riccardo GRAZZI et Massimilano PONTIL : Bilevel Programming for Hyperparameter Optimization and Meta-Learning. *In International Conference on Machine Learning (ICML)*, 2018.

[5] Quanqi HU, Yongjian ZHONG et Tianbao YANG : Multi-block Min-max Bilevel Optimization with Applications in Multi-task Deep AUC Maximization. *In Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[6] Kaiyi JI, Junjie YANG et Yingbin LIANG : Bilevel Optimization : Convergence Analysis and Enhanced Design. *In International Conference on Machine Learning (ICML)*, 2021.

[7] Rie JOHNSON et Tong ZHANG : Accelerating stochastic gradient descent using predictive variance reduction. *In Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[8] Junyi LI, Bin GU et Heng 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.

[9] Thomas MOREAU, Mathurin MASSIAS, Alexandre GRAMFORT, Pierre ABLIN, Pierre-Antoine Bannier Benjamin CHARLIER, Mathieu DAGRÉOU, Tom Dupré LA TOUR, Ghislain DURIF, Cassio F. DANTAS, Quentin KLOPFENSTEIN, Johan LARSSON, En LAI, Tanguy LEFORT, Benoit MALÉZIEUX, Badr MOUFAD, Binh T. NGUYEN, Alain RAKOTOMAMONJY, Zaccharie RAMZI, Joseph SALMON et Samuel VAITER : Benchopt : Reproducible, efficient and collaborative optimization benchmarks. *In Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[10] Lam M. NGUYEN, Jie LIU, Katya SCHEINBERG et Martin TAKÁČ : SARAH : A Novel Method for Machine Learning Problems Using Stochastic Recursive Gradient. *In International Conference on Machine Learning (ICML)*, 2017.

[11] Lam M. NGUYEN, Marten VAN DIJK, Dzung T. PHAN, Phuong Ha NGUYEN, Tsui-Wei WENG et Jayant R. KALAGNANAM : Finite-sum smooth optimization with SARAH. *Computational Optimization and Applications*, 82(3):561–593, 2022.

[12] Barak A. PEARLMUTTER : Fast Exact Multiplication by the Hessian. *Neural Computation*, 6(1):147–160, 1994.

[13] Fabian PEDREGOSA : Hyperparameter optimization with approximate gradient. *In International Conference on Machine Learning (ICML)*, 2016.

[14] Cédric ROMMEL, Thomas MOREAU, Joseph PAILLARD et Alexandre GRAMFORT : CADDA : Class-wise Automatic Differentiable Data Augmentation for EEG Signals. *In International Conference on Learning Representations (ICLR)*, 2022.

[15] Junjie YANG, Kaiyi JI et Yingbin LIANG : Provably Faster Algorithms for Bilevel Optimization. *In Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[16] Dongruo ZHOU et Quanquan GU : Lower Bounds for Smooth Nonconvex Finite-Sum Optimization. *In International Conference on Machine Learning (ICML)*, 2019.