Algorithmes stochastiques et réduction de variance grâce à un nouveau cadre pour l'optimisation bi-niveaux

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

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

[2]LAMSADE, CNRS - Place du Maréchal de Lattre de Tassigny, 75775 Paris Cedex 16, France

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

mathieu.dagreou@inria.fr, pierre.ablin@dauphine.psl.eu
samuel.vaiter@math.cnrs.fr, thomas.moreau@inria.fr

Résumé – L'optimisation bi-niveaux a de nombreuses applications en apprentissage automatique. Cet article présente un nouveau cadre pour la résolution de tels problèmes. Ce cadre permet d'éviter les différents obstacles qui se posent pour le calcul efficace du gradient de la fonction objectif. Il rend possible l'utilisation de n'importe quel estimateur non biaisé de directions de descente. Nous proposons une adaptation de l'algorithme de la descente de gradient stochastique et de l'algorithme SAGA à notre cadre. Des expériences numériques valident l'efficacité de l'approche proposée.

Abstract – Bilelvel optimization has many applications in machine learning. This paper presents a new framework to solve this kind of problems. This framework allows to avoid the bottlenecks in the computation of the gradient of the objective function, allowing the many unbiaised descent direction estimators. We propose an adaptation of the stochastic gradient descent algorithm and the SAGA algorithm to our framework and we illustrate numerically their efficiency.

1 Introduction

L'optimisation bi-niveaux attire de plus en plus l'attention dans la communauté de l'apprentissage automatique grâce à ses nombreuses applications *e.g.* la sélection d'hyperparamètres [10], la recherche d'architecture de réseaux de neurones [9], l'augmentation automatique de données [12] ou l'apprentissage profond implicite [2]. L'optimisation bi-niveaux consiste à minimiser une fonction qui dépend du résultat d'un autre problème d'optimisation, *i.e.* :

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

où F et G sont des fonctions à valeurs réelles définies sur $\mathbb{R}^p \times \mathbb{R}^d$. G est appelée la *fonction interne*, F est la *fonction externe* et h la *fonction valeur*. De même, z est la *variable interne* et x la *variable externe*. La plupart du temps, la fonction z^* peut seulement être approximée par un algorithme d'optimisation, ce qui rend l'optimisation bi-niveaux difficile d'un point de vue pratique et théorique. Sous des hypothèses appropriées, la fonction h est différentiable et son gradient est donné pour tout $x \in \mathbb{R}^d$ par

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

où $v^*(x) \in \mathbb{R}^p$ est solution d'un système linéaire

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

Dans le cadre de cet article, on suppose que F est différentiable et que pour tout x, $G(\,\cdot\,, x)$ est fortement convexe et deux fois différentiable. Ces dernières conditions sur G assurent l'existence et l'unicité de $z^*(x)$ et la validité de l'expression du gradient de h (2), qui est une conséquence du théorème des fonctions implicites appliqué à la condition d'optimalité du premier ordre sur G. Au regard de (2) et (3), le calcul du gradient de h à chaque itération est très coûteux car il suppose la résolution d'un problème d'optimisation pour avoir $z^*(x)$ et d'un système linéaire pour avoir $v^*(x)$. Cela rend l'implémentation de méthodes du premier ordre telles que la descente de gradient inefficace. On suppose par ailleurs que F et G sont des moyennes empiriques :

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

Cette structure suggère l'utilisation de méthodes stochastiques pour résoudre (1) comme la descente de gradient stochastique ou ses variantes. Cependant, comme ∇h fait intervenir l'inverse de la Hessienne de G, concevoir un estimateur non biaisé de ∇h est difficile.

Plusieurs approches ont déjà été proposées dans la littérature pour construire des méthodes stochastiques pour (1). Une première approche consiste à faire à chaque itération quelques pas de gradient stochastique et d'utiliser des approximations de Neumann stochastiques pour avoir une estimation de $v^*(x)$ [5,

8]. D'autres auteurs proposent de remplacer les approximations de Neumann par quelques pas de descente de gradient stochastique sur la fonction $v \mapsto \frac{1}{2}\langle \nabla^2_{11} G(z,x)v, v\rangle + \langle \nabla_1 F(z,x), v\rangle$ dont $v^*(x)$ est un minimiseur [1]. Enfin, une seconde famille d'approches consiste à alterner les mises à jour des variables z et x en ne faisant à chaque fois qu'un seul pas de gradient stochastique [6, 13, 7]. Nous proposons un nouveau cadre pour l'optimisation bi-niveaux en Section 2 où la variable intérieure z, la solution du système linéaire (3) v et la variable extérieure x évoluent en même temps. Les directions dans lesquelles évoluent ces variables s'écrivent comme des sommes dérivées des f_j et g_i, ce qui permet d'en déduire des estimateurs non biaisés simplement. Dans ce cadre, on propose SOBA, une adaptation du gradient stochastique et SABA, une adaptation de l'algorithme SAGA. Finalement, en Section 3, on propose une illustration numérique sur le problème de sélection d'hyperparamètres pour la régression logistique avec pénalité ℓ^2.

2 Algorithme stochastique pour l'optimisation bi-niveaux

On voudrait mettre à jour la variable x en se déplaçant dans la direction du gradient de h. Cependant, le calcul de $z^*(x)$ et $v^*(x)$ pour obtenir ce gradient est particulièrement coûteux. Plutôt que de calculer ces deux quantités exactement, on propose de suivre également des directions permettant de s'en approcher. Plus précisément, lorsque x est fixé, on approxime $z^*(x)$ en faisant un pas dans la direction du gradient $-\nabla_1 G(z,x)$. Lorsque z et x sont fixés, on approxime $v^*(x)$ en faisant un pas dans la direction $-(\nabla^2_{11} G(z,x)v + \nabla_1 F(z,x))$ qui correspond à un pas de gradient sur la fonction $v \mapsto \frac{1}{2}\langle \nabla^2_{11} G(z,x)v, v\rangle + \langle \nabla_1 F(z,x), v\rangle$. En résumé, on propose de suivre les directions suivantes :

$$D_z(z,v,x) = \nabla_1 G(z,x) \ , \tag{4a}$$
$$D_v(z,v,x) = \nabla^2_{11} G(z,x)v + \nabla_1 F(z,x) \ , \tag{4b}$$
$$D_x(z,v,x) = \nabla^2_{21} G(z,x)v + \nabla_2 F(z,x) \ . \tag{4c}$$

Celles-ci sont motivées par le fait que

$$\nabla h(x) = D_x(z^*(x), v^*(x), x) \ . \tag{5}$$

où $z^*(x)$ est le minimiseur de $G(\,\cdot\,, x)$ et $v^*(x)$ est solution de $\nabla^2_{11} G(z^*(x), x)v = -\nabla_1 F(z^*(x), x)$.

On peut noter que le calcul de ces directions ne nécessite pas le calcul des matrices $\nabla^2_{11} G(z,x)$ et $\nabla^2_{21} G(z,x)$. On utilise seulement leur produit avec des vecteurs, ce qui a un coût similaire à celui du calcul d'un gradient.

Une autre remarque importante est que ces directions sont linéaires par rapport à F et à G, ce qui n'est pas le cas de ∇h.

Algorithm 1 Cadre général

Input : initialisations $z_0 \in \mathbb{R}^p$, $x_0 \in \mathbb{R}^d$, $v_0 \in \mathbb{R}^p$, nombre d'itérations T, suite de pas $(\rho^t)_{t<T}$ et $(\gamma^t)_{t<T}$.
for $t = 0, \dots, T-1$ **do**
  Mettre à jour z : $z^{t+1} = z^t - \rho^t D_z^t$,
  Mettre à jour v : $v^{t+1} = v^t - \rho^t D_v^t$,
  Mettre à jour x : $x^{t+1} = x^t - \gamma^t D_x^t$,
  où D_z^t, D_v^t et D_x^t sont des estimateurs non biaisés de $D_z(z^t, v^t, x^t)$, $D_v(z^t, v^t, v^t)$ et $D_x(z^t, v^t, x^t)$.
end for

Cela nous permet de les écrire comme sommes :

$$D_z = \frac{1}{n}\sum_{i=1}^{n} \nabla_1 g_i(z,x) \ , \tag{6a}$$
$$D_v = \frac{1}{n}\sum_{i=1}^{n} \nabla^2_{11} g_i(z,x)v + \frac{1}{m}\sum_{j=1}^{m} \nabla_1 f_j(z,x) \ , \tag{6b}$$
$$D_x = \frac{1}{n}\sum_{i=1}^{n} \nabla^2_{21} g_i(z,x)v + \frac{1}{m}\sum_{j=1}^{m} \nabla_2 f_j(z,x) \ . \tag{6c}$$

Cela rend facile l'estimation de ces directions à partir de seulement quelques échantillons.

Fort de ces observations, on propose un cadre général pour l'optimisation bi-niveaux stochastique qui est résumé dans l'Algorithme 1 où D_z^t, D_v^t et D_x^t sont respectivement des estimateurs non-biaisés de $D_z(z^t, v^t, x^t)$, $D_v(z^t, v^t, x^t)$ et $D_x(z^t, v^t, x^t)$. On propose dans la suite deux exemples de tels estimateurs inspirés de la descente de gradient stochastique et de l'algorithme SAGA, un algorithme stochastique qui met en œuvre une technique de réduction de variance améliorant les performances.

SOBA. L'exemple le plus simple d'estimateur est obtenu en remplaçant chaque moyenne par un de ses termes choisi uniformément tel que fait classiquement dans la descente de gradient stochastique [11]. L'algorithme résultant est appelé SOBA (*StOchastic Bilevel Algorithm*). Pour ce faire, on choisit deux indices indépendants $i \in \{1, \dots, n\}$ et $j \in \{1, \dots, m\}$ et on estime chaque terme venant de G en utilisant g_i et chaque terme venant de F en utilisant f_j. Cela donne les directions non biaisées suivantes

$$D_z^t = \nabla_1 g_i(z^t, x^t) \ , \tag{7a}$$
$$D_v^t = \nabla^2_{11} g_i(z^t, x^t)v^t + \nabla_1 f_j(z^t, x^t) \ , \tag{7b}$$
$$D_x^t = \nabla^2_{21} g_i(z^t, x^t)v^t + \nabla_2 f_j(z^t, x^t) \ . \tag{7c}$$

L'algorithme SOBA consiste donc à injecter les Équations (7a) à (7c) dans l'Algorithme 1.

Une remarque importante est que toutes les directions sont calculées au même point (z^t, v^t, x^t). Cela permet de mettre à jour les variables de façon parallèle plutôt que séquentielle et donc de bénéficier d'accélération matérielle. Aussi, cela permet de partager les calculs en commun entre les différentes directions en évitant de les dupliquer. Pour des modèles plus compliqués où l'utilisation de la différentiation automatique est nécessaire, on peut stocker le graphe computationnel une seule

fois et calculer en même temps $\nabla_1 g_i(z,x)$, $\nabla^2_{11} g_i(z,x)v$ et $\nabla^2_{21} g_i(z,x)v$ en utilisant seulement une seule rétropopagation.

On donne ici un résultat de convergence théorique pour SOBA qui est démontré dans [3]. Il faut auparavant formuler quelques hypothèses sur les fonctions F et G.

Hypothèse 1. *La fonction F est différentiable et le gradient de F est Lipschitz.*

Cette hypothèse est typiquement vérifiée lorsque F est la perte des moindres carrés ou la perte logistique.

Hypothèse 2. *La fonction G est deux fois différentiable, fortement convexe par rapport à x, et que ses dérivées premières et secondes sont Lipschitz.*

La convexité forte de G et le caractère Lipschitz de son gradient sont vérifiés dans le cas de moindres carrés ou de régression logistique régularisés avec des matrices de design de rang plein.

Hypothèse 3. *La fonction $\nabla_1 F(z^*(\cdot), \cdot)$ est bornée.*

Cette hypothèse est vérifiée par exemple pour le problème de sélection d'hyperparamètres pour la régression de Ridge. Il est également nécessaire de contrôler la variance des estimateurs des différentes directions.

Hypothèse 4. *Les directions D_z^t, D_v^t et D_x^t vérifient $\mathbb{E}[\|D_z^t\|^2] \leq B_z^2(1 + D_z(z^t, v^t, x^t))$, $\mathbb{E}[\|D_v^t\|^2] \leq B_v^2(\mathbb{E}[\|z^t - z^*(x^t)\|^2 + \|v^t - v^*(x^t)\|^2])$ et $\mathbb{E}[\|D_v^t\|^2] \leq B_x^2$ pour des constantes B_z, B_v, $B_x > 0$.*

On peut alors formuler le résultat de convergence suivant.

Théorème 1. *On suppose les Hyptohèses 1 à 4 vérfiées. Pour un horizon $T > 1$ donné suffisamment grand, en prenant $\rho^t = O(T^{-\frac{2}{5}})$ et $\gamma^t = O(T^{-\frac{3}{5}})$, la suite des itérés $(x^t)_{1 \leq t \leq T}$ de SOBA vérifie*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}[\|\nabla h(x^t)\|^2] = O(T^{-\frac{2}{5}}) \ .$$

Ce taux est semblable à celui de la méthode proposé dans [6]. On a cependant une meilleure complexité grâce à l'introduction de la variable v.

SABA. Dans les problèmes d'optimisation classique, la descente de gradient stochastique à pas fixe ne peut converger à cause de la variance de l'estimation du gradient. Pour résoudre ces problèmes, des techniques de réduction de variance sont employées tel que dans l'algorithme SAGA [4].

Dans notre cadre, les directions D_z, D_v et D_x étant linéaires par rapport à F et G, il est facile d'adapter l'algorithme SAGA. L'extension que l'on propose s'appelle SABA (*Stochastic Average Bilevel Algorithm*). Notons $y = (z, v, x)$ le vecteur des variables jointes. On considère deux variables "mémoire" w_i^t pour $i \in \{1, \dots, n\}$ et $\tilde{w}_j^t$ pour $j \in \{1, \dots, m\}$. À l'itération t on tire deux indices $i \in \{1, \dots, n\}$ et $j \in \{1, \dots, m\}$ aléatoirement et on pose $w_i^{t+1} = y^t$, $\tilde{w}_j^t = y^t$ et pour $i' \neq i$ et $j' \neq j$, on pose $w_{i'}^{t+1} = w_{i'}^t$ et $\tilde{w}_{j'}^{t+1} = \tilde{w}_{j'}^t$. Chaque somme d'une direction D est approximée de la manière suivante. Étant données n fonctions $(\phi_{i'})_{i' \in \{1,\dots,n\}}$, on définit

$$S[\phi]^t = \phi_i[w_i^{t+1}] - \phi_i[w_i^t] + \frac{1}{n}\sum_{i'=1}^{n} \phi[w_{i'}^t] \ ,$$

et de même, étant données m fonctions $(\tilde{\phi}_{j'})_{j' \in \{1,\dots,m\}}$:

$$\tilde{S}[\tilde{\phi}]^t = \tilde{\phi}_j[\tilde{w}_j^{t+1}] - \tilde{\phi}_i[\tilde{w}_j^t] + \frac{1}{m}\sum_{j'=1}^{m} \tilde{\phi}[\tilde{w}_{j'}^t] \ .$$

Ces quantités sont respectivement des estimateurs non biaisés de la moyenne des $\phi_{i'}$ et des $\tilde{\phi}_{j'}$.

Avec un léger abus de notation, notons $\nabla^2_{11} Gv$ la suite de fonctions $(y \mapsto \nabla^2_{11} g_i(z,x)v)_{i \in \{1,\dots,n\}}$ et $\nabla^2_{21} Gv$ la suite de fonctions $(y \mapsto \nabla^2_{21} g_i(z,x)v)_{i \in \{1,\dots,n\}}$. On définit les **directions SABA** par

$$D_z^t = S[\nabla_1 G]^t \ , \tag{8a}$$

$$D_v^t = S[\nabla^2_{11} Gv]^t + \tilde{S}[\nabla_1 F] \ , \tag{8b}$$

$$D_x^t = S[\nabla^2_{21} Gv]^t + \tilde{S}[\nabla_2 F] \ . \tag{8c}$$

Ce sont des estimateurs non biaisés des directions D_z, D_v et D_x. L'algorithme SABA correspond à l'Algorithme 1 où l'on injecte les Équations (8a) à (8c). En pratique, les quantités $\phi_i(w_i^t)$ sont stockées plutôt que recalculées. Le coût pour calculer les directions SABA est donc le même que pour SOBA. Il requiert une mémoire supplémentaire pour les cinq quantités de taille totale $n \times p + (n+m) \times (p+d)$ flottants. Ce coût mémoire peut être réduit dans certains cas, par exemple, lorsque F et G correspondent à des modèles linéaires, dans lesquels les gradients individuels et les produits Hessienne-vecteurs sont proportionnels aux échantillons.

L'analyse théorique de SABA nécessite l'hypothèse supplémentaire suivante :

Hypothèse 5. *Pour tout $i \in \{1, \dots, n\}$ et $j \in \{1, \dots, m\}$, les fonctions ∇g_i, ∇f_j, $\nabla^2_{11} g_i$ et $\nabla^2_{21} g_i$ sont Lipschitz en (z, x).*

On fournit le résultat théorique suivant qui montre que le taux de convergence de SABA est le même que celui de la descente de gradient. La preuve de ce résultat est dans [3].

Théorème 2. *On suppose vérifiées les Hypothèses 1 à 3 ainsi que l'Hypothèse 5. En prenant des pas constants ρ et γ suffisamment petits, on a*

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

Ce taux de convergence est, à notre connaissance, le meilleur parmi les algorithmes stochastiques pour les problèmes bi-niveaux.

3 Évaluation numérique

On compare les performances de SOBA et SABA avec d'autres méthodes concurrentes en sélection d'hyperparamètres. Les méthodes comparées sont stocBiO [8], AmIGO [1], MRBO [13], TTSA [6], BSA [5] et SUSTAIN. [7].

Chaque méthode a un pas interne et externe. Pour chaque algorithme, on effectue une recherche par grille pour trouver la paire de pas qui donne la convergence la plus rapide. Chaque pas a un taux de décroissance donné par la théorie : par exemple pour SOBA, les pas sont $\rho^t = \alpha t^{-\frac{5}{2}}$ et $\gamma^t = \beta t^{-\frac{3}{5}}$, où α et β sont choisis dans une recherche par grille. Pour SABA, $\rho^t = \alpha$ et $\gamma^t = \beta$, où α et β sont choisis dans une recherche par grille. Cette procédure est répétée pour chaque algorithme. La paire (α, β) est à chaque fois choisie parmi 63 pairs possibles. Chaque expérience est répétée 10 fois et la médiane et les quantiles entre 20% et 80% sont affichés.

La tâche à laquelle on s'attèle est la sélection du paramètre de régularisation pour une régression logistique avec pénalité ℓ^2. Notons $((d_i^{\text{train}}, y_i^{\text{train}}))_{1\leq i\leq n}$ et $((d_i^{\text{val}}, y_i^{\text{val}}))_{1\leq i\leq m}$ les ensembles d'entraînement et de validation. Dans ce cas, la variable interne θ correspond aux paramètres du modèle et la variable externe λ correspond à la régularisation. Les fonctions F et G du problème (1) sont la perte logistique avec pénalité ℓ^2 pour G, c'est-à-dire

$$F(\theta, \lambda) = \frac{1}{m}\sum_{i=1}^{m} \varphi(y_i^{\text{val}} \langle d_i^{\text{val}}, \theta\rangle) \text{ , et} \tag{9}$$

$$G(\theta, \lambda) = \frac{1}{n}\sum_{i=1}^{n} \varphi(y_i^{\text{train}} \langle d_i^{\text{train}}, \theta\rangle) + \frac{1}{2} R(\theta, \lambda) \tag{10}$$

où $R(\theta, \lambda) = \theta^\top \mathbf{diag}(e^{\lambda_1}, \dots, e^{\lambda_p})\theta$ et $\varphi(u) = \log(1 + e^{-u})$. On entraîne un classifieur binaire sur le jeu de données IJCNN1 [1]. Ce jeu contient $n = 49\,990$ échantillons d'entraînement, $m = 91\,701$ échantillons de validation et on a $p = 22$.

On affiche en Figure 1 l'écart de sous-optimalité $h(\lambda^t) - h(\lambda^*)$ pour chacune des méthodes. Les valeurs les plus basses sont atteintes par AmIGO et SABA, mais plus rapidement par SABA. SABA est la seule méthode basée sur une seule boucle dont la sous-optimalité arrive en-dessous de 10^{-3}. Parmi les autres méthodes, SOBA est la première à atteindre son plateau, mais ce plateau est le plus haut. L'écart entre SOBA et SABA montre le gain de la réduction de variance : cela donne un plateau plus bas avec des pas fixés.

4 Conclusion

On a présenté ici un nouveau cadre pour l'optimisation bi-niveaux qui permet de développer facilement des algorithmes stochastiques. L'avantage de notre méthode est que les directions dans les Équations (4a) à (4c) sont linéaires par rapport à G et F. On propose alors SOBA et SABA, des extensions des algorithmes d'optimisation stochastiques classiques. Ce cadre ouvre un grand champ de potentielles méthodes pour l'optimisation stochastique pour les problèmes bi-niveaux faisant intervenir des techniques d'extrapolation, de réduction de variance, de moments etc.

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

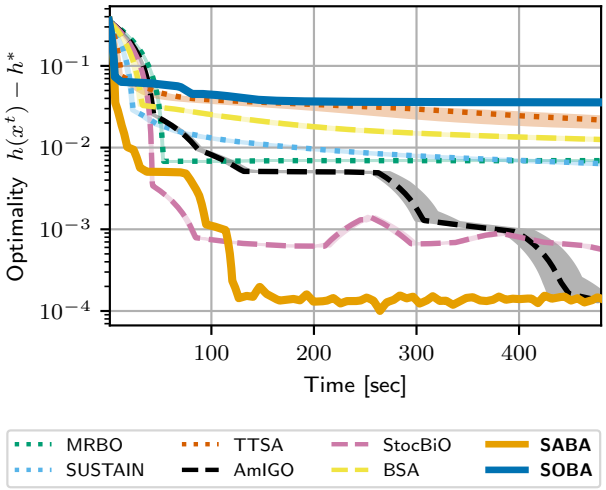

FIGURE 1 – Comparaison de SOBA et SABA avec d'autres méthodes stochastiques pour l'optimisation bi-niveaux sur de la sélection d'hyperparamètres pour régression logistique avec pénalisation ℓ^2 sur le jeu de données IJCNN1. Pour chaque méthode, on affiche la performance médiane sur 10 essais. On voit que SABA obtient les meilleures performances.

Références

[1] M. Arbel, J. Mairal. Amortized Implicit Differentiation for Stochastic Bilevel Optimization. In *International Conference on Learning Representations (ICLR)*, 2022.

[2] S. Bai, J. Z. Kolter, V. Koltun. Deep Equilibrium Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[3] M. Dagréou, P. Ablin, S. Vaiter, T. Moreau. A framework for bilevel optimization that enables stochastic and global variance reduction algorithms. *arXiv 2201.13409*, 2022.

[4] A. Defazio, F. Bach, 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.

[5] S. Ghadimi, M. Wang. Approximation Methods for Bilevel Programming. *arXiv 1802.02246*, 2018.

[6] M. Hong, M.-T. Wai, Z. Wang, Z. Yang. A Two-Timescale Framework for Bilevel Optimization : Complexity Analysis and Application to Actor-Critic. *arXiv 2007.05170*, 2021.

[7] P. Khanduri, S. Zeng, M. Hong, H.-T. Wai, Z. Wang, Z. Yang. A Near-Optimal Algorithm for Stochastic Bilevel Optimization via Double-Momentum. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[8] K. Ji, J. Yang, Y. Liang. Bilevel optimization : Convergence analysis and enhanced design. In *International Conference on Machine Learning (ICML)*, 2021.

[9] H. Liu, K. Simonyan, Y. Yang. Darts : Differentiable ar-chitecture search. In *International Conference on Learn-ing Representations (ICLR)*, 2018.

[10] F. Pedregosa. Hyperparameter optimization with approximate gradient. In *International Conference on MachineLearning (ICML)*, 2016.

[11] H. Robbins, S. Monro. A stochastic approximation method. In *The Annals of Mathematical Statistics*, 22(3) :400–407, 1951.

[12] C. Rommel, T. Moreau, J. Paillard, A. Gramfort. CADDA : Class-wise Automatic Differentiable Data Augmentation for EEG Signals, In *International Conference on Learning Representations (ICLR)*, 2022.

[13] J. Yang, K. Ji, Y. Liang. Provably Faster Algorithms for Bilevel Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.