Pré-Publicações do Departamento de Matemática
Universidade de Coimbra
Preprint Number 21–53

BILEVEL STOCHASTIC METHODS FOR OPTIMIZATION AND MACHINE LEARNING: BILEVEL STOCHASTIC DESCENT AND DARTS

T. GIOVANNELLI, G. KENT AND L. N. VICENTE

Abstract: Two-level stochastic optimization formulations have become instrumental in a number of machine learning contexts such as neural architecture search, continual learning, adversarial learning, and hyperparameter tuning. Practical stochastic bilevel optimization problems become challenging in optimization or learning scenarios where the number of variables is high or there are constraints.

The goal of this paper is twofold. First, we aim at promoting the use of bilevel optimization in large-scale learning and we introduce a practical bilevel stochastic gradient method (BSG-1) that requires neither lower level second-order derivatives nor system solves (and dismisses any matrix-vector products). Our BSG-1 method is close to first-order principles, which allows it to achieve a performance better than those that are not, such as DARTS. Second, we develop bilevel stochastic gradient descent for bilevel problems with lower level constraints, and we introduce a convergence theory that covers the unconstrained and constrained cases and abstracts as much as possible from the specifics of the bilevel gradient calculation.

Keywords: Bilevel optimization, machine learning, stochastic gradient descent, DARTS.
Math. Subject Classification (2010): 90C06, 90C15, 90C26, 90C90.

1. Introduction

Many real-world applications are naturally formulated using hierarchical objectives, which are organized into different nested levels. In the bilevel case, the main goal is placed into an upper optimization level, while the lower optimization level aims to determine the best response to a decision made in the upper level. Bilevel optimization has a rich literature of algorithmic development and theory (see [1, 6, 9, 44, 46] for extensive surveys and books on this topic). The main applications are found in game theory, defense industry, and optimal structural design, and one has recently seen a surge of contributions to machine learning (see, e.g., [13, 25], and the recent review [26]).

Received December 06, 2021.
Support for LNV was partially provided by the Centre for Mathematics of the University of Coimbra under grant FCT/MCTES UIDB/MAT/00324/2020.

In this paper, we consider the following nonlinear bilevel optimization problem (BLP) formulation

$$\begin{aligned} \min_{x\in\mathbb{R}^n,\, y\in\mathbb{R}^m} \ & f_u(x,y) \\ \text{s.t. } & x \in X \\ & y \in \operatorname*{argmin}_{y\in Y(x)} \ f_\ell(x,y). \end{aligned} \tag{BLP}$$

The goal of the upper level (UL) problem is to determine the optimal value of the UL function $f_u : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, where the UL variables x are subjected to UL constraints ($x \in X$) and the UL variables y are subjected to being an optimal solution of the lower level (LL) problem. In the LL problem, the LL function $f_\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is optimized in the LL variables y, subject to the LL constraints $y \in Y(x)$. Since the goal of this paper is to propose and analyze a general optimization methodology for a stochastic BLP, the LL problem is assumed to be well-defined, in the sense of having a unique solution $y(x)$ for all $x \in X$. Hence, problem BLP is equivalent to a problem posed solely in the UL variables:

$$\min_{x\in\mathbb{R}^n} \ f(x) = f_u(x, y(x)) \quad \text{s.t.} \quad x \in X. \tag{1.1}$$

Also, note that the UL constraints ($x \in X$) are only posed in the UL variables x as otherwise problem BLP could become intractable in the sense of having a disconnected feasible region in the (x, y)–space. Whenever applying orthogonal projections within stochastic gradient type methods, the sets X and $Y(x)$ will be assumed closed and convex.

Under appropriate smoothness and non-singularity assumptions, the gradient of f at $(x, y(x))$, when $Y(x) = \mathbb{R}^m$, is given by the so-called adjoint formula

$$\nabla f \ = \ \nabla_x f_u - \nabla^2_{xy} f_\ell (\nabla^2_{yy} f_\ell)^{-1} \nabla_y f_u, \tag{1.2}$$

where all gradients and Hessians are evaluated at $(x, y(x))$. The adjoint gradient can be computed by first solving the adjoint equation $\nabla^2_{yy} f_\ell \, \lambda = -\nabla_y f_u$ for the adjoint variables $\lambda = \lambda(x, y(x))$, and then calculating $\nabla_x f_u + \nabla^2_{xy} f_\ell \, \lambda$. One arrives at the adjoint formula by applying the chain rule to $f_u(x, y(x))$ and calculating the Jacobian of $y(x)$ through the (sensitivity) equations $\nabla_y f_\ell(x, y(x)) = 0$.

When $Y(x) \neq \mathbb{R}^m$, it is no longer possible to explicitly apply an adjoint principle, but one can compute the steepest descent direction for f through

an auxiliary linear-quadratic bilevel problem [42] grounded on sensitivity principles (see Subsection 2.2).

1.1. Bilevel machine learning. A variety of problems arising in machine learning can be formulated in terms of bilevel optimization problems: continual learning, neural architecture search, adversarial training, and hyperparameter tuning are among the most popular examples (see [26] for a review on this topic).

Continual Learning (CL) aims to train ML models when the static task usually considered in learning problems (classification, regression, etc.) is replaced by a sequence of tasks that become available one at a time [29], and for which training and validation datasets are increasingly larger. For each task, a CL instance is formulated as a bilevel problem, where at the UL problem one minimizes the validation error on a subset of model parameters (which includes all hyperparameters), and at the LL problem the training error is minimized on the remaining parameters. A sequence of bilevel problems is then solved for each consecutive task. In a sense, CL is close to meta-learning [21], where the goal is to determine the best learning process. The increasing interest in CL is motivated by the demand for approaches that help neural networks to learn new tasks without forgetting the previous ones, a phenomenon which is referred to as *catastrophic forgetting* [14, 18, 32].

Another relevant ML area where bilevel optimization is instrumental is Neural Architecture Search (NAS) for Deep Learning. The goal of this problem is to automate the task of designing Deep Neural Networks (DNNs) such that the network?s prediction error is minimized. In recent years, NAS was proposed in a bilevel optimization formulation [25] that represents a continuous relaxation over the (discrete) architecture search space such that gradient information can be utilized. In this way, the UL problem consists of minimizing the validation error of the network over the architecture space and the LL problem consists of minimizing the training error with respect to the network weights. Designing these structures is a vital component of any ML task that utilizes DNNs and typically requires substantial effort through trial and error on behalf of human experts. As a result, automating this design phase in an efficient and optimal way is a critical issue at the heart of the Deep Learning community. For a thorough discussion on the different approaches that have been proposed for NAS, see the survey [12].

Finally, two other popular classes of ML problems that can be formulated by using bilevel optimization are adversarial training and hyperparameter tuning. Adversarial training aims to robustly address adversarial examples [45] which cannot be correctly classified by ML models once a small perturbation is applied. The adversarial training problem is handled by solving a min-max problem [23], but any such problem can be reformulated as a bilevel one. The max/LL problem is posed on the variables which perturb the data in a worst-case fashion, where the UL/min problem attempts to minimize the training error on the ML model parameters [17, 31]. Hyperparameter tuning aims to find the best value for the hyperparameters used in a ML model in order to increase its *generalization capability* [2, 10, 13]. In the bilevel formulations proposed in the literature, the UL problem optimizes the validation error over the hyperparameters, while the LL problem has the goal of finding the NN weights that minimize the training error.

1.2. Bilevel stochastic descent. In bilevel stochastic optimization, f_u and f_ℓ can be interpreted as expected values, namely, $f_u = \mathbb{E}[f_u(x, y, w_u)]$ and $f_\ell = \mathbb{E}[f_\ell(x, y, w_\ell)]$, where w_u and w_ℓ are random variables defined in a probability space (with probability measure independent from x and y) such that i.i.d. samples can be observed or generated. The same applies to the functions possibly defining $Y(x)$. (To keep notation simple, we are using the same f_u and f_ℓ for deterministic and random variants.)

Having in mind ML applications such as NAS and CL, the methods we are considering are Stochastic Approximation (SA) techniques, of the type of the stochastic gradient (SG) method [5, 39, 40] for single-objective optimization. In fact, the bilevel stochastic gradient (BSG) can be seen as a SG method applied to (1.1), $x_{k+1} = x_k - \alpha_k g_k^{\mathrm{BSG}}$, where α_k is the stepsize or learning rate, and g_k^{BSG} is a stochastic gradient of f, obtained by sampling the gradients and Hessians in (1.2) at $(x_k, \tilde{y}_k)$. The stochastic gradient g_k^{BSG} may be inexact when $\tilde{y}_k \neq y(x_k)$.

In the bilevel optimization literature, stochastic gradient-based methods are commonly classified according to the approach used to compute the second term in the adjoint formula (1.2). In particular, a first category is composed of the algorithms that either solve the adjoint equation [35] or use a truncated Neumann series to approximate the inverse of the Hessian $\nabla^2_{yy} f_\ell$ [30], while a second category includes all the approaches based on automatic differentiation [13]. Note that our BSG-1 method falls into the first

L. N. Vicente
Department of Industrial and Systems Engineering, Lehigh University, 200 West Packer Avenue, Bethlehem, PA 18015-1582, USA and Centre for Mathematics of the University of Coimbra (CMUC), Portugal (lnv@mat.uc.pt).

[37] S. N. Ravi, T. Dinh, V. S. Lokhande, and V. Singh. Explicitly imposing constraints in deep networks via conditional gradients gives improved generalization and faster convergence. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4772–4779, 2019.

[38] P. Ren, Y. Xiao, X. Chang, P. Huang, Z. Li, X. Chen, and X. Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Comput. Surv.*, 54, 2021.

[39] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[40] J. Sacks. Asymptotic distribution of stochastic approximation procedures. *Annals of Mathematical Statistics*, 29:373 – 405, 1958.

[41] S. Sangalli, E. Erdil, A. Hoetker, O. Donati, and E. Konukoglu. Constrained optimization to train neural networks on critical and under-represented Classes. *arXiv e-prints*, page arXiv:2102.12894, February 2021.

[42] G. Savard and J. Gauvin. The steepest descent direction for the nonlinear bilevel programming problem. *Operations Research Letters*, 15:265–272, 1994.

[43] A. Shaker, F. Alesiani, S. Yu, and W. Yin. Bilevel continual learning, 2020.

[44] A. Sinha, P. Malo, and K. Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22:276–295, 2018.

[45] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv e-prints*, page arXiv:1312.6199, December 2013.

[46] L. N. Vicente and P. H. Calamai. Bilevel and multilevel programming: A bibliography review. *Journal of Global Optimization*, 5:291–306, 1994.

[47] J. Yang, K. Ji, and Y. Liang. Provably Faster Algorithms for Bilevel Optimization. *arXiv e-prints*, page arXiv:2106.04692, June 2021.

T. Giovannelli
Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015-1582, USA (tog220@lehigh.edu).

G. Kent
Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015-1582, USA (gdk220@lehigh.edu).

class of algorithms, which in the numerical experiments reported in [19] is observed to have better performance. The use of such algorithms for unconstrained bilevel optimization has been promoted in [7, 8, 16, 20, 22]; see also [26] for a recent review. In [7], the BSG method has been applied to hyperparameter tuning in SVM using the adjoint formula, which is simplified using the structure of the problem. The authors [8] provided a convergence analysis of the general BSG method but requiring the LL problem to be solved to optimality at each iteration. In [16], both deterministic and stochastic algorithms are proposed, and their convergence analysis is developed without requiring an exact solution of the LL problem at each iteration. The gradient of f was computed using the adjoint formula, approximating the inverse of the Hessian $\nabla^2_{yy} f_\ell$ by a truncated Neumann series. An algorithm with better complexity bounds than [16] is presented in [22], which also shows that computing the adjoint gradient by using automatic differentiation may be less computationally efficient than solving the adjoint equation or using the truncated Neumann series. Finally, in [20] the authors study how the convergence results of their algorithm depend on the relative size of the UL and LL stepsizes, and they adapt their approach for a bilevel problem in reinforcement learning.

DARTS [25] is an optimization technique related to the BSG method, which has enjoyed great popularity in NAS. It always considers an inexact solution to the LL problem, and it starts an iteration by making it less inexact by applying one step of SG to the LL problem, $\tilde{y}_k = y_k - \eta \nabla_y f_\ell(x_k, y_k)$. Then, it displaces the UL variables using $x_{k+1} = x_k - \eta\, g_k^{\rm DARTS}$, where η is a fixed stepsize, and $g_k^{\rm DARTS}$ is computed by applying the chain rule to $\nabla_x f_u(x, y - \eta \nabla_y f_\ell(x, y))$, leading to

$$g_k^{\rm DARTS} \;=\; \nabla_x f_u(x_k, \tilde{y}_k) - \eta \nabla^2_{xy} f_\ell(x_k, y_k) \nabla_y f_u(x_k, \tilde{y}_k). \tag{1.3}$$

However, $g_k^{\rm DARTS}$ may not be a descent direction, even in the deterministic case, and a direct comparison with $g_k^{\rm BSG}$ reveals the lack of LL curvature, meaning the use of $\nabla^2_{yy} f_\ell(x_k, y_k)$. The matrix-vector product in (1.3) is approximated by finite differences (see Subsection 5.2), rendering DARTS free of both second-order derivatives and matrix-vector products.

1.3. Contributions of the paper. A major difficulty in the adjoint formula (1.2) is the use of second-order derivatives of f_ℓ and the need to solve the adjoint equation $\nabla^2_{yy} f_\ell\, \lambda = -\nabla_y f_u$, which prevents its application to

large-scale ML problems. A first main contribution of this paper is the idea of approximating the second-order derivatives by using the outer product of the corresponding gradients, i.e.,

$$\nabla^2_{xy} f_\ell \;\simeq\; \nabla_x f_\ell \nabla_y f_\ell^\top \;\text{ and }\; \nabla^2_{yy} f_\ell \simeq \nabla_y f_\ell \nabla_y f_\ell^\top.$$

The above approximations are inspired by Gauss-Newton (GN) methods for nonlinear least-squares problems (see, e.g,. [34]), where the Hessian matrix of the objective function $\sum_{i=1}^p (r_i - a_i)^2$ (in which each r_i is a scalar function and a_i a scalar) is approximated by $\sum_{i=1}^p \nabla r_i \nabla r_i^\top$, and also from the fact that the empirical risk of misclassification in ML is often a sum of non-negative terms matching a function to a scalar which can then be considered in a least-squares fashion [3, 15]. The resulting approximate adjoint equation $(\nabla_y f_\ell \nabla_y f_\ell^\top)\,\lambda = -\nabla_y f_u$ is most likely infeasible, and we suggest solving it in the least-squares sense. One solution is $\lambda = -\nabla_y f_u / (\nabla_y f_\ell^\top \nabla_y f_\ell)$. Plugging this and $\nabla^2_{xy} f_\ell \simeq \nabla_x f_\ell \nabla_y f_\ell^\top$ in the adjoint formula (1.2) gives rise to our practical BSG calculation

$$\nabla_x f_u - \frac{\nabla_y f_\ell^\top \nabla_y f_u}{\nabla_y f_\ell^\top \nabla_y f_\ell} \nabla_x f_\ell. \tag{1.4}$$

This approximated BSG allows us to use the adjoint formula without computing Hessians or even Hessian-vector products, which is prohibitively expensive for the large bilevel problems arising in ML. It will be referred to as BSG-1, the "1" standing for first-order rank-1 approximations of the Hessian matrices.

The numerical experiments reported in this paper show that the BSG-1 method performs significantly better than DARTS. A byproduct contribution of this paper is then precisely a number of ways of potentially improving DARTS by drawing a comparison between the g_k^{BSG} and g_k^{DARTS} expressions, and by bringing the new ideas of the BSG-1 gradient approximation (1.4). One can scale the term $\nabla_y f_u(x_k, \tilde{y}_k)$ in g_k^{DARTS} by $1/\|\nabla_y f_\ell(x_k, \tilde{y}_k)\|^2$, thus including the missing curvature $(\nabla^2_{yy} f_\ell)^{-1}$ of the LL objective function. One can also consider taking $\eta = 1$ in g_k^{DARTS} (regardless of the value assigned to η when updating the LL variables). If we went a step further and address $\nabla^2_{xy} f_\ell$ in DARTS as in the BSG-1 method, the two methods would be essentially the same, with a minor difference being whether the evaluations are done at (x_k, y_k) or at $(x_k, \tilde{y}_k)$. There is thus a road-map from one method to the

[25] H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable architecture search. *ArXiv*, arXiv:1806.09055, 2019.

[26] R. Liu, J. Gao, J. Zhang, D. Meng, and Z. Lin. Investigating bi-Level optimization for learning and vision from a unified perspective: A survey and beyond. *arXiv e-prints*, page arXiv:2101.11517, January 2021.

[27] S. Liu and L. N. Vicente. The stochastic multi-gradient algorithm for multi-objective optimization and its application to supervised machine learning. *arXiv e-prints*, page arXiv:1907.04472, July 2019.

[28] S. Liu and L. N. Vicente. Accuracy and fairness trade-offs in machine learning: A stochastic multi-objective approach. *arXiv e-prints*, page arXiv:2008.01132, August 2020.

[29] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. *arXiv e-prints*, page arXiv:1706.08840, June 2017.

[30] J. Lorraine, P. Vicol, and D. Duvenaud. Optimizing Millions of Hyperparameters by Implicit Differentiation. *arXiv e-prints*, page arXiv:1911.02590, November 2019.

[31] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[32] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of *Psychology of Learning and Motivation*, pages 109–165. Academic Press, 1989.

[33] A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19:1574–1609, 2009.

[34] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, Berlin, second edition, 2006.

[35] F. Pedregosa. Hyperparameter optimization with approximate gradient. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 737–746, New York, New York, USA, 20–22 Jun 2016. PMLR.

[36] Q. Pham, D. Sahoo, C. Liu, and S. C. H. Hoi. Bilevel continual learning, 2020.

[12] T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

[13] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1568–1577. PMLR, 2018.

[14] R. M. French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3:128–135, 1999.

[15] M. Gargiani, A. Zanelli, M. Diehl, and F. Hutter. On the promise of the stochastic generalized Gauss-Newton method for training DNNs. *arXiv e-prints*, page arXiv:2006.02409, June 2020.

[16] S. Ghadimi and M. Wang. Approximation methods for bilevel programming, 2018.

[17] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[18] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv e-prints*, page arXiv:1312.6211, December 2013.

[19] R. Grazzi, L. Franceschi, M. Pontil, and S. Salzo. On the iteration complexity of hypergradient computation. *arXiv e-prints*, page arXiv:2006.16218, June 2020.

[20] M. Hong, H. Wai, Z. Wang, and Z. Yang. A Two-Timescale Framework for Bilevel Optimization: Complexity Analysis and Application to Actor-Critic. *arXiv e-prints*, page arXiv:2007.05170, July 2020.

[21] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey. Meta-Learning in neural networks: A survey. *arXiv e-prints*, page arXiv:2004.05439, April 2020.

[22] K. Ji, J. Yang, and Y. Liang. Bilevel Optimization: Convergence Analysis and Enhanced Design. *arXiv e-prints*, page arXiv:2010.07962, October 2020.

[23] H. Jiang, Z. Chen, Y. Shi, B. Dai, and T. Zhao. Learning to defend by learning to attack. *arXiv e-prints*, page arXiv:1811.01213, November 2018.

[24] R Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. KDD'96, page 202–207. AAAI Press, 1996.

other which can give practitioners different computational tools for solving bilevel ML problems.

Another main contribution of this paper is a general convergence theory for BSG that is as much as possible abstracted from the specifics of the adjoint formula (1.2) for the gradient of f, in particular in what regards the inverse of the Hessian of f_ℓ and/or the solution of the adjoint system. Our theory is grounded on sensitivity principles of nonlinear optimization and extends naturally to the constrained LL case ($Y(x) \neq \mathbb{R}^m$), which has not been covered elsewhere, neither algorithmically nor theoretically.

1.4. Organization of this paper. This paper is organized as follows. In Section 2, we describe the BSG method for both the unconstrained and constrained lower-level cases. The convergence analysis of the method, along with the assumptions required, is reported in Section 3. Section 4 focuses on continual learning and its formulation as a bilevel optimization problem. Numerical results for this problem and for a synthetic logistic regression instance are analyzed in Section 5, which also describes the BSG-1 and DARTS methods used for benchmarking purposes. Finally, in Section 6 we draw some concluding remarks and we propose ideas for future work. By default, all norms $\|\cdot\|$ used in this paper are the ℓ_2 ones.

2. The bilevel stochastic gradient method

In this section, we introduce the bilevel stochastic gradient (BSG) method for solving stochastic BLPs. Let $\{w_k^u\}_{k\geq 0}$ and $\{w_k^\ell\}_{k\geq 0}$ be sequences of random variables for UL and LL function evaluations, respectively. A realization of the random variables w_k^u and w_k^ℓ can be interpreted as a single sample or a batch of samples for mini-batch SG use. For compactness of notation, let us set $\xi_k = (w_k^u, w_k^\ell)$ for the combined UL and LL random variables.

The schema of the Bilevel Stochastic Gradient (BSG) method is presented in Algorithm 1. An initial point (x_0, y_0) and a sequence of positive scalars $\{\alpha_k\}$ are required as input. In Step 1, any arbitrary optimization method can be applied to approximately solve the LL problem, regardless of being unconstrained ($Y(x) = \mathbb{R}^m$) or constrained ($Y(x) \neq \mathbb{R}^m$). In Step 2, one computes an approximated (negative) BSG, which will be denoted by $-d(x_k, \tilde{y}_k, \xi_k)$ and will be detailed in the next two subsections. Finally, at Step 3, the vector x is updated by using a proper step size taken from the sequence of positive scalars. When X is a closed and convex constrained set different from

$\mathbb{R}^n$, we need to compute the orthogonal projection of $x_k + \alpha_k\, d(x_k, \tilde{y}_k, \xi_k)$ onto X (note that such a projection can be computed by solving a convex optimization problem).

Algorithm 1 Bilevel Stochastic Gradient (BSG) Method

Input: $(x_0, y_0) \in \mathbb{R}^n \times \mathbb{R}^m$, $\{\alpha_k\}_{k\geq 0} > 0$.

For $k = 0, 1, 2, \ldots$ **do**

Step 1. Obtain an approximation $\tilde{y}_k$ to the LL optimal solution $y(x_k)$.

Step 2. Obtain an approximation $d(x_k, \tilde{y}_k, \xi_k)$ to $-g_k^{\text{BSG}}$.

Step 3. Compute $x_{k+1} = P_X(x_k + \alpha_k\, d(x_k, \tilde{y}_k, \xi_k))$.

End do

We point out that as is usual in the literature related to SG methods, a stopping criterion is not considered due to a lack of reasonable criteria and for the need to study the asymptotic convergence properties.

2.1. The unconstrained lower level case. Given $(x_k, \tilde{y}_k)$, we denote by $g_x^u(x_k, \tilde{y}_k, w_k^u)$, $g_y^u(x_k, \tilde{y}_k, w_k^u)$, and $g_y^\ell(x_k, \tilde{y}_k, w_k^\ell)$ the stochastic gradient estimates that approximate $\nabla_x f_u(x_k, \tilde{y}_k)$, $\nabla_y f_u(x_k, \tilde{y}_k)$, and $\nabla_y f_\ell(x_k, \tilde{y}_k)$, respectively. The same notation applies to the stochastic Hessian estimates: $H_{xy}^\ell(x_k, \tilde{y}_k, w_k^\ell)$ and $H_{yy}^\ell(x_k, \tilde{y}_k, w_k^\ell)$ approximate, respectively, $\nabla_{xy}^2 f_\ell(x_k, \tilde{y}_k)$ and $\nabla_{yy}^2 f_\ell(x_k, \tilde{y}_k)$.

In the unconstrained LL case ($Y(x) = \mathbb{R}^m$), an approximated (negative) BSG for use in Step 2 can be computed directly from the adjoint formula (1.2), as follows:

$$\begin{aligned} -d(x_k, \tilde{y}_k, \xi_k) \;&= g_x^u(x_k, \tilde{y}_k, w_k^u) \\ &\quad - H_{xy}^\ell(x_k, \tilde{y}_k, w_k^\ell)(H_{yy}^\ell(x_k, \tilde{y}_k, w_k^\ell))^{-1} g_y^u(x_k, \tilde{y}_k, w_k^u). \end{aligned} \tag{2.1}$$

The theory of Section 3 will cover (2.1). The data in the formula (1.2) is referred to as $D(x, y)$ or $D(x, y(x))$, depending on the point where the gradients and Hessians are evaluated:

$$D(x, y) \;=\; \left(\nabla_x f_u(x, y), \nabla_y f_u(x, y), \nabla_{xy}^2 f_\ell(x, y), \nabla_{yy}^2 f_\ell(x, y)\right). \tag{2.2}$$

is crucial to allow the application of the BSG method to NAS, which would not be possible otherwise due to the extreme dimensions of the resulting bilevel problems. Moreover, the fact that BSG can solve bilevel optimization problems with constrained LL problems paves the way for the solution of new NAS formulations. In particular, one could think of including in the LL problem constraints that help the model avoid overfitting [37] or constraints that depend on the specific learning instances considered [41].

References

[1] J. F. Bard. *Practical Bilevel Optimization: Algorithms and Applications*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[2] K. P. Bennett, G. Kunapuli, J. Hu, and J. S. Pang. *Bilevel Optimization and Machine Learning*, pages 25–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[3] A. Botev, H. Ritter, and D. Barber. Practical Gauss-Newton optimisation for deep learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 557–565. PMLR, 06–11 Aug 2017.

[4] T. Chen, Y. Sun, and W. Yin. A Single-Timescale Stochastic Bilevel Optimization Method. *arXiv e-prints*, page arXiv:2102.04671, February 2021.

[5] K. L. Chung. On a stochastic approximation method. *Annals of Mathematical Statistics*, 25:463 – 483, 1954.

[6] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153:235–256, 2007.

[7] N. Couellan and W. Wang. Bi-level stochastic gradient for large scale support vector machine. *Neurocomputing*, 153:300–308, 2015.

[8] N. Couellan and W. Wang. On the convergence of stochastic bi-level gradient methods. *PREPRINT available at http://www.optimization-online.org/DB_HTML/2016/02/5323.html*, 02 2016.

[9] S. Dempe and A. Zemkoho. *Bilevel Optimization: Advances and Next Challenges*. Springer International Publishing, 2020.

[10] S. Dhar, U. Kurup, and M. Shah. Stabilizing bi-Level hyperparameter optimization using Moreau-Yosida regularization. *arXiv e-prints*, page arXiv:2007.13322, July 2020.

[11] D. Dua and C. Graff. UCI machine learning repository, 2017.

problems due to its practical satisfactory performance. Our numerical experiments showed that BSG-1 performs significantly better than DARTS on the logistic regression and continual learning instances considered. The modifications we suggested for improving DARTS are effective when tested on the logistic regression instances. Further exploration of improvements for BSG-1 and DARTS methods on other large-scale learning problems is left as future work (in particular the use of variance reduction techniques, as already proposed in [4, 47]).

Moreover, BSG methods have been so far limited to the unconstrained LL case because the only resource used has been the adjoint formula. It is however possible to calculate a descent direction in the constrained LL case, and we have thus shown how to develop BSG methods for such a case. An approximate BSG direction can be obtained through the solution of an auxiliary LQ bilevel problem; see Subsection 2.2. Our convergence theory already indicates that if the error in the norm between such a direction and the negative gradient of f is of the order of the stepsize, one retains the classical sublinear SG rate. One needs to numerical test the constrained LL approach presented in this paper and make it efficient, robust, and scalable. Finally, the algorithmic framework and the convergence theory have to be further developed to address the non-convex case (at both upper and lower levels) as well as UL constraints in both UL and LL variables.

Significant innovation is expected when BSG is applied to the formulations that are proposed to overcome the critical issue of *catastrophic forgetting* in continual learning. The formulation introduced in [29] uses inequality constraints to ensure that, at each stage, the current model outperforms the old model on the previous tasks, thus preventing the deterioration of the classification accuracy when learning new tasks. A similar idea is exploited in the bilevel formulation proposed in [43], where the violation of the constraints in [29] is penalized (and hence such constraints are possibly not satisfied at the optimal solution). One expects a considerable improvement by adapting problem (4.1) to the constrained LL case, and then solving it using the BSG direction described in Subsection 2.2.

The promising results on the ML instances considered in this proposal suggest that the BSG-1 method has the potential to perform well on the unconstrained bilevel formulations of NAS, which in the literature are still tackled by using DARTS when a continuous relaxation of the (discrete) search space is used [38]. We point out that using the rank-1 Hessian approximations

The data in the calculation (2.1) is referred to as $D(x_k, \tilde{y}_k, \xi_k)$:

$$D(x_k, \tilde{y}_k, \xi_k) \; = \\ \left(g_x^u(x_k, \tilde{y}_k, w_k^u), g_y^u(x_k, \tilde{y}_k, w_k^u), H_{xy}^\ell(x_k, \tilde{y}_k, w_k^\ell), H_{yy}^\ell(x_k, \tilde{y}_k, w_k^\ell)\right). \tag{2.3}$$

As we have said before, our practical implementation of BSG will use rank-one approximations for $H_{xy}^\ell(x_k, \tilde{y}_k, w_k^\ell)$ and $H_{yy}^\ell(x_k, \tilde{y}_k, w_k^\ell)$, and then solve $H_{yy}^\ell(x_k, \tilde{y}_k, w_k^\ell)\lambda = -g_y^u(x_k, \tilde{y}_k, w_k^u)$ in the least-squares sense.

2.2. The constrained lower level case. Let us now handle the constrained LL case, in which we consider

$$Y(x) \; = \; \{f_i(x,y) = 0, \; i \in I, \;\; f_i(x,y) \leq 0, \; i \in J\},$$

where I and J are two finite sets of indices. For the LL problem, assume that at $y(x)$, the gradients of the active constraints are linearly independent (LICQ) and the sufficient second-order optimality conditions are satisfied [34]. Let $L_\ell(x, y; z) = f_\ell(x, y) + \sum_{I(x)\cup J} z_i f_i(x, y)$ be the Lagrangian function, where $I(x)$ includes all the active indices and z are the Lagrange multipliers. The unique multipliers associated with $y(x)$ under LICQ are denoted by $z(x)$.

The steepest descent direction $d(x, y(x)) \in \mathbb{R}^n$ for f at x can be calculated by solving the following linear-quadratic bilevel problem [42]:

$$\begin{aligned}
\min_{d^x\in\mathbb{R}^n,\, d^y\in\mathbb{R}^m} \; & \nabla_x f_u(x, y(x))^\top d^x + \nabla_y f_u(x, y(x))^\top d^y \\
\text{s.t. } & \|d^x\|_\infty \leq 1, \\
& d^y \in \operatorname*{argmin}_{d^y\in\mathbb{R}^m} (d^x, d^y)^\top \nabla^2 L_\ell(x, y(x); z(x))(d^x, d^y) \\
& \qquad \text{s.t. } \nabla_y f_\ell(x, y(x))^\top d^y = -\nabla_x f_\ell(x, y(x))^\top d^x \\
& \qquad\qquad\qquad + \nabla_x L_\ell(x, y(x); z(x))^\top d^x, \\
& \qquad\qquad \nabla_y f_i(x, y(x))^\top d^y \leq -\nabla_x f_i(x, y(x))^\top d^x, \quad i \in I(x), \\
& \qquad\qquad \nabla_y f_i(x, y(x))^\top d^y = -\nabla_x f_i(x, y(x))^\top d^x, \quad i \in J. \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{LQ}(x, y(x))
\end{aligned}$$

Note that problem LQ(x, y) can be solved for any pair of variables (x, y), where y is not necessarily an optimal solution $y(x)$ to the LL problem of BLP. In such a case, we denote the linear-quadratic problem as LQ(x, y) and its optimal solution as $d(x, y)$, while the problem data is referred to as $D(x, y)$.

It is important to point out that when y is not the LL optimal solution $y(x)$, the direction obtained by solving the problem $\mathrm{LQ}(x, y)$ is not guaranteed to be descent. Also note that the UL problem of this linear-quadratic bilevel problem constrains d^x in the ℓ_∞-norm, and thus $d(x, y(x))$ might not necessarily recover $-\nabla f(x)$, although the two directions are co-linear. One points out that in the case $Y(x) = \mathbb{R}^m$, there are no constrains in the LL problem of $\mathrm{LQ}(x, y(x))$, and the Hessian of its objective function involves only the second-order derivatives of f_ℓ. A simple calculation allows us to conclude that indeed $d(x, y(x))$ recovers a positive multiple of the adjoint gradient in (1.2) (normalized in the ℓ_∞-norm).

In the stochastic version of $\mathrm{LQ}(x_k, y(x_k))$, we replace $\nabla_x f_u(x_k, y(x_k))$, $\nabla_y f_u(x_k, y(x_k))$, $\nabla_y f_\ell(x_k, y(x_k))$, and $\nabla^2 L_\ell(x_k, y(x_k); z(x_k))$ by the corresponding stochastic gradient and Hessian estimates at $(x_k, \tilde{y}_k)$, where $\tilde{y}_k$ is not necessarily $y(x_k)$. A solution of the resulting linear-quadratic bilevel problem, $\mathrm{LQ}(x_k, \tilde{y}_k, \xi_k)$, is denoted by $d(x_k, \tilde{y}_k, \xi_k)$. Such a solution is not guaranteed to yield a descent direction when $\tilde{y}_k = y(x_k)$, as opposed to its deterministic counterpart $d(x_k, y(x_k))$. The data of problem $\mathrm{LQ}(x_k, \tilde{y}_k, \xi_k)$ is referred to as $D(x_k, \tilde{y}_k, \xi_k)$.

In practical terms it is important to point out that problem $\mathrm{LQ}(x_k, \tilde{y}_k, \xi_k)$ can be approximately solved by solving an LP relaxation, which consists of replacing its LL problem (a QP) by its first-order necessary conditions (which are linear in d^x and d^y as long as there are no inequality constraints).

3. Convergence rate of the BSG method in the strongly convex case

In this section, we extend the convergence theory of the SG method to the bilevel case when the stepsize is assumed to be decaying. The BSG method under analysis considers an inexact solution of the LL problem. The BSG is not required to follow a specific adjunct calculation, and the constrained LL setting is also comprehensively covered. The BLP objective function f is assumed to be strongly convex (leading to a $1/k$ sublinear convergence rate) or simply convex ($1/\sqrt{k}$ rate).

3.1. General assumptions. Again, let us recall that $f(x) = f_u(x, y(x))$. The true function f will later be assumed sufficiently smooth (Assumption 3.6 below). For the moment we need to impose a certain smoothness of the BLP

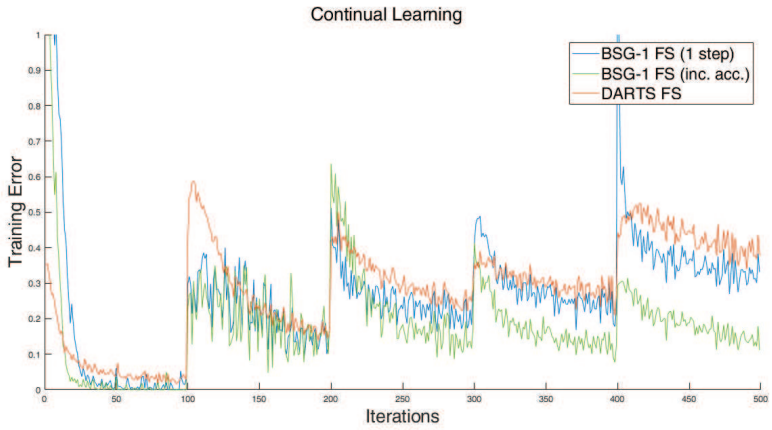

FIGURE 3. Comparison of two variants of the BSG-1 method (1 step and inc. acc.) with the DARTS one when fixed stepsizes (FS) (of value equal to 0.007) are used. The comparison is based on the number of iterations.

and LL fixed stepsizes of 0.007 for the three algorithmic variants such that the results would be comparable. We emphasize that DARTS cannot make further improvements by taking more steps in the LL problem, whereas BSG-1 will perform better the more accurately the LL problem is solved, leaving room for even further improvement in performance.

6. Concluding remarks and future work

In this paper, we introduced a practical bilevel stochastic gradient (BSG) method for large-scale bilevel optimization problems, we suggested simple and effective improvements for DARTS, and we provided a general convergence theory for BSG methods that also covers the constrained LL case. Even in the unconstrained LL case, the convergence theory presented in this paper allows for more abstraction from the building blocks of adjoint gradients.

In particular, the use of rank-1 Hessian approximations in our BSG-1 method allowed us to both approximate the second-order derivatives in the adjoint formula and avoid explicitly solving the adjoint equation (also dismissing any matrix-vector products). Although DARTS does not incorporate descent or first-order principles, it is commonly applied to solve NAS

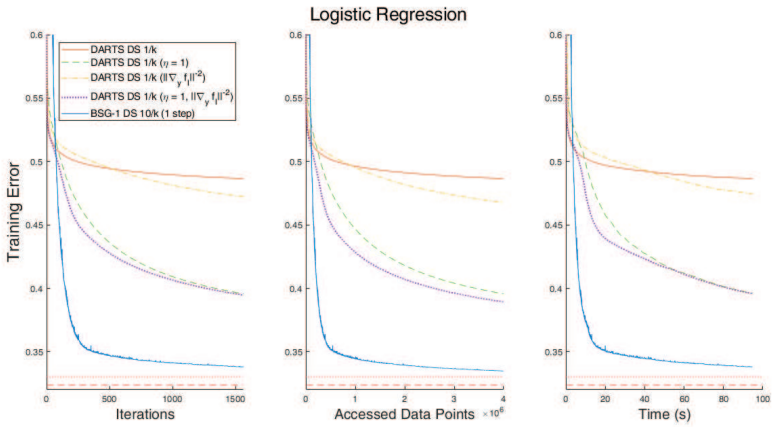

FIGURE 2. Comparison of four different variants of the DARTS method with the BSG-1 one. The mini-batch size is 512 for both the UL and LL problems.

Further, we implemented a DNN with two convolutional layers and one linear layer as our model. The network consisted of 18,816 and 125,450 weights in the hidden and output layers, respectively. For the UL and LL problems, we have used batches of sizes equal to 0.5% and 0.1% of the ones of the current task's validation and training datasets, respectively. When evaluating the UL and LL objective functions, such percentages are increased to 5% and 1%, respectively, to ensure better accuracy.

The results from the numerical experiment can be seen in Figure 3, where two variants of BSG-1 are compared against DARTS when fixed stepsizes (FS) are used. In particular, for BSG-1, such results were gathered when using in the LL problem either 1 step or an increasing accuracy strategy with threshold equal to 10^{-1}. By the nature of the CL problem, we expect to see five separate "jumps" in the validation error indicating the start of a new task. As the tasks progress, both variants of BSG-1 converge faster than DARTS, as they achieve smaller values of the validation error, especially the BSG variant using the increasing accuracy strategy. DARTS is not able to guarantee a sufficiently small error on the validation data for all tasks, despite some initial improvement obtained. For this problem, we used UL

gradients and Hessians involved in the computation of the BSG direction (Assumption 3.1 below).

Assumption 3.1 (Smoothness of gradients and Hessians for the BSG direction)**.** *All gradients and Hessians involved in the computation of the BSG direction are Lipschitz continuous in* (x, y).

In the unconstrained LL case, and when using the adjoint formula (1.2), Assumption 3.1 amounts to assuming that $\nabla_x f_u$, $\nabla_y f_u$, $\nabla^2_{xy} f_\ell$, and $\nabla^2_{yy} f_\ell$ are Lipschitz continuous in y. In the constrained LL case (LQ subproblem), this is extended to $\nabla_x f_\ell$, $\nabla_y f_\ell$, $\nabla_x f_i$, $\nabla_y f_i$, $\nabla^2_{xy} f_i$, and $\nabla^2_{yy} f_i$, for all $i \in I \cup J$.

We require the approximate BSG direction $d(x_k, \tilde{y}_k, \xi_k)$ as well as the true steepest descent direction $d(x_k, y(x_k))$ to satisfy Assumption 3.2. The expected value with respect to $\xi_k = (w^u_k, w^\ell_k)$ is denoted by $\mathbb{E}_{\xi_k}[\cdot]$.

Assumption 3.2. *The vectors* $d(x_k, y(x_k))$ *and* $d(x_k, \tilde{y}_k, \xi_k)$ *satisfy the following conditions*

(i) *There exists a positive scalar* C_d *such that*

$$\mathbb{E}_{\xi_k}[\|d(x_k, y(x_k)) - (-\nabla f(x_k))\|] \;\leq\; C_d\, \alpha_k. \tag{3.1}$$

(ii) *There exists a positive scalar* G_d *such that*

$$\mathbb{E}_{\xi_k}[\|d(x_k, \tilde{y}_k, \xi_k)\|^2] \;\leq\; G_d. \tag{3.2}$$

In the unconstrained LL case, $d(x_k, y(x_k)) = -\nabla f(x_k)$ is trivially satisfied. In the constrained LL case we know that $d(x_k, y(x_k))$ is co-linear with $-\nabla f(x_k)$ but their norms may differ, and thus the need for this assumption.

Note that instead of assuming (3.2), we could have assumed $\mathbb{V}_{\xi_k}[d(x_k, \tilde{y}_k, \xi_k)] \leq G_V$ and $\mathbb{E}_{\xi_k}[d(x_k, \tilde{y}_k, \xi_k)] \leq G_E$, for some positive constants G_V, G_E. In fact, from the definition of variance, $\mathbb{E}_{\xi_k}[\|d(x_k, \tilde{y}_k, \xi_k)\|^2] \leq G_V + G_E^2 = G_d$. Also, in the constrained LL case, a bound on the second moment of $d(x_k, \tilde{y}_k, \xi_k)$ comes directly from the constraint $\|d^x\|_\infty \leq 1$ in problem $\mathrm{LQ}(x_k, \tilde{y}_k, \xi_k)$, which then implies $\mathbb{E}_{\xi_k}[\|d(x_k, \tilde{y}_k, \xi_k)\|^2] \leq n$.

3.2. Sensitivity of the approximated bilevel stochastic gradient direction. To bound the expectation of the error between the true gradient or steepest descent direction $d(x_k, y(x_k))$ and the approximate BSG direction $d(x_k, \tilde{y}_k, \xi_k)$, we will need to apply sensitivity analysis arguments from nonlinear optimization. In particular, we need to assume that the calculation process of the BSG direction is Lipschitz continuous with respect to changes in its data.

Assumption 3.3 (Sensitivity of the BSG direction)**.** *The calculation of the BSG direction $d(D)$ is a Lipschitz continuous function of the data, i.e., there exists a $L_{BSG} > 0$ such that, for any D_1 and D_2, one has*

$$\|d(D_1) - d(D_2)\| \;\leq\; L_{BSG}\|D_1 - D_2\|.$$

In the unconstrained LL case, when the adjoint formula (1.2) or (2.1) is used, the data D is either the deterministic one (2.2) or the stochastic one (2.3), respectively. Assumption 3.3 amounts to assuming that all gradient vectors and Hessian matrices involved are bounded and that $\nabla^2_{yy} f_\ell$ and H^ℓ_{yy} are bounded away from singularity. In fact, sum is Lipschitz continuous, multiplication is Lipschitz continuous if the factors are bounded, and symmetric matrix inversion is also Lipschitz continuous if its eigenvalues are bounded from zero. For the adjoint gradient calculation, we have $d(a, b, A, B) = a - AB^{-1}b$, with $D = (a, b, A, B)$, and one can easily prove that

$$\|d(D_1) - d(D_2)\| \;\leq\; C\left(\|a_1 - a_2\| + \|b_1 - b_2\| + \|A_1 - A_2\| + \|B_1 - B_2\|\right), \tag{3.3}$$

where C depends on a bound on b, A, and B^{-1}, and the right-hand side is a norm on (a, b, A, B).

We now introduce Assumption 3.4, which can be easily enforced in practice by solving the LL problem with the SG method (see Subsection 3.5).

Assumption 3.4. *There exists a positive scalar C_y such that*

$$\mathbb{E}_{\xi_k}[\|y(x_k) - \tilde{y}_k\|] \;\leq\; C_y\,\alpha_k. \tag{3.4}$$

Finally, we need Assumption 3.5 below to hold which essentially amounts to the sampling error in the data. To enforce this assumption in practice, we refer the reader to the discussion reported in Section 3.6. Recall that $D(x_k, \tilde{y}_k)$ represents the data defining the true quantity $d(x_k, \tilde{y}_k)$, and $D(x_k, \tilde{y}_k, \xi_k)$ the data of the calculation of $d(x_k, \tilde{y}_k, \xi_k)$. In the unconstrained LL case, $D(x_k, \tilde{y}_k)$ and $D(x_k, \tilde{y}_k, \xi_k)$ are given by (2.2) and (2.3), respectively.

Assumption 3.5. *There exists a positive scalar C_D such that*

$$\mathbb{E}_{\xi_k}[\|D(x_k, \tilde{y}_k) - D(x_k, \tilde{y}_k, \xi_k)\|] \;\leq\; C_D\,\alpha_k.$$

One is ready to establish the desired error bound in terms of α_k, for which the constant will depend on the above-introduced constants L_{BSG}, C_y, and C_D.

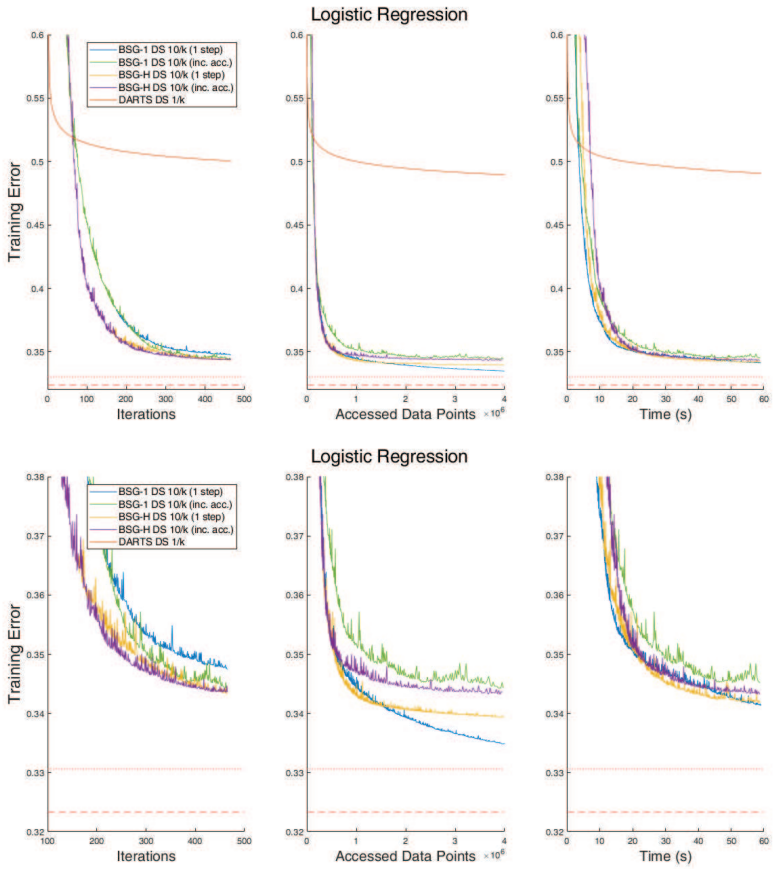

FIGURE 1. Comparison of four different variants of the BSG method with the DARTS one (the lower plots zoom in for the BSG ones). The mini-batch size is 512 for both the UL and LL problems.

datasets (D^2_{val} and D^2_{train}) consist of the images with class labels in $\{0, 1, 2, 3\}$, etc., until the final task datasets (D^5_{val} and D^5_{train}), which are the original training and validation sets and consist of all the class labels $\{0, 1, ..., 9\}$.

From Figure 1, we can see that all BSG variants perform significantly better than DARTS. The BSG versions equipped with the LL inc. acc. (with threshold equal to 10^{-4}) yield the smallest prediction error in the least number of iterations. However, this requires accessing a large number of samples. The most cost-effective performance in terms of accessed data points is exhibited by the BSG versions equipped with the LL 1 step, especially the BSG-1.

Considering the stronger performance achieved by BSG-1, we performed additional testing to study possible ways of improving DARTS, as discussed in Subsection 1.3. In particular, we assessed the performance of DARTS when scaling the term $\nabla_y f_u(x_k, \tilde{y}_k)$ in g_k^{DARTS} by the factor $1/\|\nabla_y f_\ell(x_k, \tilde{y}_k)\|^2$ (which in the plots is referred to as $\|\nabla_y f_\ell\|^{-2}$) and when taking $\eta = 1$ in g_k^{DARTS} regardless of the value assigned to η for updating the LL variables (we indicate this modification as $\eta = 1$). Figure 2 shows that on the logistic regression instance considered, DARTS significantly improves in both cases, and the improvement is even larger when both modifications are considered together.

The red dotted horizontal lines in both figures represent the minimum value of the true function f (recall that $f(x) = f_u(x, y(x))$), found by the full-batch/deterministic BSG method with line-search. The red dashed horizontal lines in both figures mark the minimum value of the upper level function f_u (in x and y) when relaxing the original problem by suppressing the LL problem, found by MATLAB with `fmincon`.

5.4. Results for continual learning instances. We now present numerical results comparing BSG-1 and DARTS on the CL problems (4.1) that were posed in Section 4. In our implementation, we determine λ_t on the current problem by starting from the parameter values found from the previous problem. However, since each consecutive task increases the output space of the DNN, we entirely re-initialize δ_t at the start of each new task so that the model outputs are not biased from previous tasks.

In order to test our algorithm on a large-scale ML scenario, we chose the well-studied MNIST dataset that consists of 70,000 black-and-white images (28×28) of hand-written digits. We used 60,000 images for training and the remaining 10,000 images for validation. We solved five problems (4.1) with an increasing number of tasks from 1 to 5, where the first task datasets (D^1_{val} and D^1_{train}) consist of only the images with class labels in $\{0, 1\}$, the second task

Lemma 3.1. *Under Assumptions 3.1, 3.3, 3.4 and 3.5,*

$$\mathbb{E}_{\xi_k}[\|d(x_k, y(x_k)) - d(x_k, \tilde{y}_k, \xi_k)\|] \;\leq\; B\,\alpha_k, \tag{3.5}$$

where $B = L_{BSG}(L_{LL}C_y + C_D)$, and $L_{LL} > 0$ is a constant only dependent on the Lipschitz constants of the gradients and Hessians of Assumption 3.1.

Proof: By adding and subtracting the term $d(x_k, \tilde{y}_k)$ and using the triangle inequality, we have

$$\mathbb{E}_{\xi_k}[\|d(x_k, y(x_k)) - d(x_k, \tilde{y}_k, \xi_k)\|] \;\leq\; \mathbb{E}_{\xi_k}[\|d(x_k, y(x_k)) - d(x_k, \tilde{y}_k)\|] \tag{3.6}$$

$$+\,\mathbb{E}_{\xi_k}[\|d(x_k, \tilde{y}_k) - d(x_k, \tilde{y}_k, \xi_k)\|]. \tag{3.7}$$

Now we derive a bound for the right-hand side in (3.6). By considering the data $D_1 = D(x_k, y(x_k))$ and $D_2 = D(x_k, \tilde{y}_k)$ in Assumption 3.3, we obtain

$$\|d(x_k, y(x_k)) - d(x_k, \tilde{y}_k)\| \;\leq\; L_{BSG}\|D(x_k, y(x_k)) - D(x_k, \tilde{y}_k)\|. \tag{3.8}$$

Taking the expectation on both sides of (3.8), we have

$$\mathbb{E}_{\xi_k}[\|d(x_k, y(x_k)) - d(x_k, \tilde{y}_k)\|] \;\leq\; L_{BSG}\,\mathbb{E}_{\xi_k}[\|D(x_k, y(x_k)) - D(x_k, \tilde{y}_k)\|]. \tag{3.9}$$

Note that the right-hand side of (3.9) contains exact BLP gradients and Hessians. Therefore, Lipschitz continuity of those mappings (Assumption 3.1) implies the existence of a constant $L_{LL} > 0$ such that

$$\|D(x_k, y(x_k)) - D(x_k, \tilde{y}_k)\| \;\leq\; L_{LL}\|y(x_k) - \tilde{y}_k\|. \tag{3.10}$$

Taking the expectation on both sides of (3.10), we can write

$$\mathbb{E}_{\xi_k}[\|D(x_k, y(x_k)) - D(x_k, \tilde{y}_k)\|] \;\leq\; L_{LL}\,\mathbb{E}_{\xi_k}[\|y(x_k) - \tilde{y}_k\|]. \tag{3.11}$$

Therefore, from (3.9), (3.11), and Assumption 3.4, we obtain

$$\mathbb{E}_{\xi_k}[\|d(x_k, y(x_k)) - d(x_k, \tilde{y}_k)\|] \;\leq\; L_{BSG}\,L_{LL}\,C_y\,\alpha_k. \tag{3.12}$$

Now we derive a bound for (3.7). By considering the data $D_1 = D(x_k, \tilde{y}_k)$ and $D_2 = D(x_k, \tilde{y}_k, \xi_k)$ in Assumption 3.3, we have

$$\|d(x_k, \tilde{y}_k) - d(x_k, \tilde{y}_k, \xi_k)\| \;\leq\; L_{BSG}\|D(x_k, \tilde{y}_k) - D(x_k, \tilde{y}_k, \xi_k)\|. \tag{3.13}$$

Taking the expectation on both sides of (3.13), we obtain

$$\mathbb{E}_{\xi_k}[\|d(x_k, \tilde{y}_k) - d(x_k, \tilde{y}_k, \xi_k)\|] \;\leq\; L_{BSG}\,\mathbb{E}_{\xi_k}[\|D(x_k, \tilde{y}_k) - D(x_k, \tilde{y}_k, \xi_k)\|]. \tag{3.14}$$

Hence, from (3.14) and Assumption 3.5, we obtain

$$\mathbb{E}_{\xi_k}[\|d(x_k, \tilde{y}_k) - d(x_k, \tilde{y}_k, \xi_k)\|] \;\leq\; L_{BSG}\,C_D\,\alpha_k. \tag{3.15}$$

The proof can be concluded from (3.6)–(3.7), (3.12), and (3.15). ∎

3.3. Rate in the strongly convex case. Both strongly convex and convex cases require an explicit assumption on the smoothness of the true function f.

Assumption 3.6 (Smoothness of f)**.** *The gradient ∇f is Lipschitz continuous in x with constant $L_{\nabla f} > 0$, i.e.,*

$$\|\nabla f(x) - \nabla f(\bar{x})\| \;\leq\; L_{\nabla f}\|x - \bar{x}\| \quad \text{for all } (x, \bar{x}) \in \mathbb{R}^n \times \mathbb{R}^n.$$

In the unconstrained LL case, ∇f has the explicit adjoint formula given in (1.2), and its Lipschitz continuity can be inferred from the Lipschitz continuity of $y(x)$ and of the gradients and Hessians involved and from the boundedness away from singularity of $\nabla^2_{yy} f_\ell$. The details can be easily worked out (see, for instance, [8, Lemma 2.2]) or simply by applying (3.3) followed by the Lipschitz continuity of the gradients and Hessians. In the constrained LL case, this assumption is a supposition on the sensitivity of the auxiliary subproblem $\mathrm{LQ}(x, y(x))$.

We also need the iterates to lie in a bounded set, which could be ensured by the boundedness of X in the BLP formulation.

Assumption 3.7 (Boundedness of the iterates)**.** *The sequence of iterates $\{x_k\}_{k\in\mathbb{N}}$ yielded by Algorithm 1 is contained in a bounded set.*

Assumption 3.7 implies that there exists a positive constant Θ such that, for any (k_1, k_2), we have

$$\|x_{k_1} - x_{k_2}\| \;\leq\; \Theta \;<\; \infty. \tag{3.16}$$

Finally, we assume that the true function f is strongly convex.

Assumption 3.8 (Strong convexity)**.** *The function f is strongly convex, namely, there exists a constant $c > 0$ such that*

$$f(\bar{x}) \;\geq\; f(x) + \nabla f(x)^\top(\bar{x} - x) + \frac{c}{2}\|\bar{x} - x\|^2 \text{ for all } (\bar{x}, x) \in \mathbb{R}^n \times \mathbb{R}^n. \tag{3.17}$$

The next theorem proves that under the assumption of strong convexity and decaying stepsize ($\sum_{k=1}^\infty \alpha_k = \infty$ and $\sum_{k=1}^\infty \alpha_k^2 < \infty$), the sequence of points yielded by Algorithm 1 generates a sequence of f values that decays sublinearly at the rate of $1/k$. We use $E[\cdot]$ to refer to the *total expectation* of f, namely, the expected value with respect to the joint distribution of all the random vectors ξ_k.

superset (the UL one) but guaranteeing a good performance in a subset (the LL one).

For simplicity, we will perform binary classification using a logistic regression loss for misclassification. Let us denote the whole features/labels dataset by $\mathcal{D} = \{(z_j, u_j),\, j \in \{1, \ldots, N\}\}$. The prediction model consists of a separating hyperplane $c^\top z + b$ that delivers a correct classification if

$$\begin{cases} c^\top z_j + b \geq 0 \text{ when } u_j = +1, \\ c^\top z_j + b < 0 \text{ when } u_j = -1. \end{cases}$$

To measure misclassification, we use the (smooth and convex) logistic regression loss function $\ell(z, u; c, b) = \log(1 + \exp(-u(c^\top z + b)))$.

We split the dataset into two groups of dimension N_{T_1} and N_{T_2}, respectively, which are used to train the prediction model at both levels of the BLP, which is formulated as

$$\begin{aligned} \min_{(\bar{c},\, \bar{b},\, c,\, b)} \; & \frac{1}{N_{T_1}} \sum_{j=1}^{N_{T_1}} \ell(z_j, u_j; \bar{c}, \bar{b}) + \frac{1}{N_{T_2}} \sum_{j=1}^{N_{T_2}} \ell(z_j, u_j; c, b) \\ \text{s.t. } (c,\; b) \in \operatorname*{argmin}_{(c,\; b)} \; & \frac{1}{N_{T_2}} \sum_{j=1}^{N_{T_2}} \ell(z_j, u_j; c, b) + \frac{\lambda}{2}\|(\bar{c}, \bar{b}) - (c, b)\|^2. \end{aligned} \tag{5.3}$$

We refer to the UL objective function as the prediction error. Note that $x = (c, b)$ are the UL variables, while $y = (\bar{c}, \bar{b})$ are the LL variables. The second term in the LL objective function (where λ is a positive parameter) requires the y's to be close to the x's, so that the model predicted on the superset does not deviate much from the one predicted on the subset. In the numerical experiments reported, we used the *Adult Income* dataset [24] (which can be found in the UCI Machine Learning repository [11]), after performing the preprocessing described in [28, Appendix D]. We randomly chose 37,500 instances (split into $N_{T_1} = 30,000$ and $N_{T_2} = 7,500$).

We compared BSG-1 and BSG-H with DARTS using the best decaying stepsize (DS) sequence found for both (i.e., $\{10/k\}_{k\in\mathbb{N}}$ for BSG-1/BSG-H and $\{1/k\}_{k\in\mathbb{N}}$ for DARTS). Three metrics are used for the comparison: the number of iterations, the number of accessed data points, and the CPU time. The number of accessed data points refers to the number of samples that are used in each algorithm to compute the stochastic gradients and, in the BSG-H case, also the Hessians.

difference of the UL objective function between two consecutive iterations is less than a given threshold, thus leading to an increasing accuracy strategy. In any of the two cases, we feed SG with a hotstart $\tilde{y}_k^0 = \tilde{y}_{k-1}$ (and set $\tilde{y}_{-1} = y_0$).

5.2. DARTS. DARTS was proposed in [25] for the solution of stochastic BLPs arising from NAS, and was briefly mentioned in Section 1.2. Only the unconstrained LL case ($Y(x) = \mathbb{R}^m$) has been considered. To avoid the computation of the second-order derivatives in (1.3), DARTS approximates the matrix-vector product $\nabla^2_{xy} f_\ell(x_k, y_k)\nabla_y f_u(x_k, \tilde{y}_k)$ by a finite-difference (FD) scheme [25]:

$$\nabla^2_{xy} f_\ell(x_k, y_k)\nabla_y f_u(x_k, \tilde{y}_k) \;=\; \frac{\nabla_x f_\ell(x_k, y_k^+) - \nabla_x f_\ell(x_k, y_k^-)}{2\varepsilon},$$

where

$$y_k^\pm \;=\; y_k \pm \varepsilon \nabla_y f_u(x_k, \tilde{y}_k) \quad \text{with} \quad \varepsilon \;=\; 0.01/\|\nabla_y f_u(x_k, \tilde{y}_k)\|. \tag{5.2}$$

Algorithm 3 reports the schema of DARTS for the stochastic setting. In Step 1, a single step of SG (with fixed stepsize η) is applied to the LL problem to obtain an approximation $\tilde{y}_k$ to the LL optimal solution. Then, in Step 2, the UL variables are updated by moving along the "approximated" descent direction using a fixed stepsize α.

Algorithm 3 Differentiable Architecture Search (DARTS)

Input: $(x_0, y_0) \in \mathbb{R}^n \times \mathbb{R}^m$, $\alpha > 0$, $\eta > 0$.

For $k = 0, 1, 2, \ldots$ **do**

Step 1. Compute $\tilde{y}_k = y_k - \eta\, g_y^\ell(x_k, y_k, w_k^\ell)$.

Step 2. Compute $x_{k+1} =$
$x_k - \alpha\left(g_x^u(x_k, \tilde{y}_k, w_k^u) - \frac{\eta}{2\varepsilon}(g_x^\ell(x_k, y_k^+, w_k^\ell) - g_x^\ell(x_k, y_k^-, w_k^\ell))\right)$,
with $y_k^\pm$ and ε as in (5.2), with $g_y^u(x_k, \tilde{y}_k, w_k^u)$ instead of $\nabla_y f_u(x_k, \tilde{y}_k)$, and set $y_{k+1} = \tilde{y}_{k+1}$.

End do

5.3. Results for synthetic learning instances. We first report results for a "synthetic" bilevel supervised ML problem, where one attempts to minimize the prediction error on both levels, but where the LL training set is a subset of the upper one. It is as if we wanted to learn a predictor from a

Theorem 3.2. *Let Assumptions 3.1–3.8 hold and x_* be the unique minimizer of f. Consider the schema given by Algorithm 1 and assume a decaying step size of the form $\alpha_k = \frac{2}{c(k+1)}$. The sequence of iterates yielded by Algorithm 1 satisfies*

$$\min_{s=1,\ldots,k} \mathbb{E}[f(x_s)] - \mathbb{E}[f(x_*)] \;\leq\; \frac{2G_d + 4(B + C_d)\Theta}{c(k+1)}.$$

Proof: For any $k \in \mathbb{N}$, we can write

$$\begin{aligned}
\mathbb{E}_{\xi_k}[\|x_{k+1} - x_*\|^2] &= \mathbb{E}_{\xi_k}[\|P_X(x_k + \alpha_k\, d(x_k, \tilde{y}_k, \xi_k)) - x_*\|^2] \\
&\leq \mathbb{E}_{\xi_k}[\|x_k + \alpha_k\, d(x_k, \tilde{y}_k, \xi_k) - x_*\|^2] \\
&= \|x_k - x_*\|^2 + \alpha_k^2 \mathbb{E}_{\xi_k}[\|d(x_k, \tilde{y}_k, \xi_k)\|^2] \\
&\quad + 2\alpha_k \mathbb{E}_{\xi_k}[d(x_k, \tilde{y}_k, \xi_k)]^\top (x_k - x_*).
\end{aligned}$$

Adding and subtracting the terms $2\alpha_k(\mathbb{E}[d(x_k, y(x_k))] + \nabla f(x_k))^\top (x_k - x_*)$ and applying the Cauchy-Schwarz inequality, we obtain

$$\begin{aligned}
\mathbb{E}_{\xi_k}[\|x_{k+1} - x_*\|^2] \leq\; & \|x_k - x_*\|^2 + \alpha_k^2 \mathbb{E}_{\xi_k}[\|d(x_k, \tilde{y}_k, \xi_k)\|^2] \\
& - 2\alpha_k (\nabla f(x_k))^\top (x_k - x_*) \\
& + 2\alpha_k \mathbb{E}_{\xi_k}[d(x_k, \tilde{y}_k, \xi_k) - d(x_k, y(x_k))]^\top (x_k - x_*) \\
& + 2\alpha_k \mathbb{E}_{\xi_k}[d(x_k, y(x_k)) + \nabla f(x_k)]^\top (x_k - x_*) \\
\leq\; & \|x_k - x_*\|^2 + \alpha_k^2 \mathbb{E}_{\xi_k}[\|d(x_k, \tilde{y}_k, \xi_k)\|^2] \\
& - 2\alpha_k (\nabla f(x_k))^\top (x_k - x_*) \\
& + 2\alpha_k \|\mathbb{E}_{\xi_k}[d(x_k, \tilde{y}_k, \xi_k) - d(x_k, y(x_k))]\| \|x_k - x_*\| \\
& + 2\alpha_k \|\mathbb{E}_{\xi_k}[d(x_k, y(x_k)) - (-\nabla f(x_k))]\| \|x_k - x_*\|.
\end{aligned}$$

By Jensen's inequality, the previous expression can be written as follows

$$\begin{aligned}
\mathbb{E}_{\xi_k}[\|x_{k+1} - x_*\|^2] \leq\; & \|x_k - x_*\|^2 + \alpha_k^2 \mathbb{E}_{\xi_k}[\|d(x_k, \tilde{y}_k, \xi_k)\|^2] \\
& - 2\alpha_k (\nabla f(x_k))^\top (x_k - x_*) && (3.18) \\
& + 2\alpha_k \mathbb{E}_{\xi_k}[\|d(x_k, \tilde{y}_k, \xi_k) - d(x_k, y(x_k))\|] \|x_k - x_*\| && (3.19) \\
& + 2\alpha_k \mathbb{E}_{\xi_k}[\|d(x_k, y(x_k)) - (-\nabla f(x_k))\|] \|x_k - x_*\|. && (3.20)
\end{aligned}$$

Assumption 3.8 implies that

$$\nabla f(x_k)^\top (x_k - x_*) \;\geq\; f(x_k) - f(x_*) + \frac{c}{2}\|x_k - x_*\|^2. \tag{3.21}$$

Then, by using point (i) of Assumption 3.2, Assumption 3.7, and inequalities (3.2), (3.5), and (3.21), the terms (3.18), (3.19), and (3.20) can be bounded as follows

$$\begin{aligned}\mathbb{E}_{\xi_k}[\|x_{k+1}-x_*\|^2] &\leq \|x_k-x_*\|^2 + G_d\alpha_k^2 - 2\alpha_k f(x_k) + 2\alpha_k f(x_*)\\ &\quad - c\alpha_k\|x_k-x_*\|^2\\ &\quad + 2B\Theta\alpha_k^2 + 2C_d\Theta\alpha_k^2\\ &= (1-c\alpha_k)\|x_k-x_*\|^2 + 2\alpha_k(f(x_*)-f(x_k))\\ &\quad + (G_d + 2(B+C_d)\Theta)\,\alpha_k^2.\end{aligned}$$

Denoting $M = G_d + 2(B+C_d)\Theta$, using $\alpha_k = \frac{2}{c(k+1)}$, and rearranging, we obtain

$$\begin{aligned}\mathbb{E}_{\xi_k}[f(x_k)-f(x_*)] &\leq \frac{(1-c\alpha_k)\|x_k-x_*\|^2 + \alpha_k^2 M - \mathbb{E}_{\xi_k}[\|x_{k+1}-x_*\|^2]}{2\alpha_k}\\ &= \frac{c(k-1)}{4}\|x_k-x_*\|^2 + \frac{M}{c(k+1)}\\ &\quad - \frac{c(k+1)}{4}\mathbb{E}_{\xi_k}[\|x_{k+1}-x_*\|^2].\end{aligned}$$

If we replace k by s, we take the total expectation, we multiply both sides by s, and we sum over $s = 1,\ldots,k$, we obtain

$$\begin{aligned}\sum_{s=1}^{k} s\,\mathbb{E}[f(x_s)-f(x_*)] &\leq \sum_{s=1}^{k}\left(\frac{cs(s-1)}{4}\mathbb{E}[\|x_s-x_*\|^2]\right.\\ &\quad \left. - \frac{cs(s+1)}{4}\mathbb{E}[\|x_{s+1}-x_*\|^2]\right) + \sum_{s=1}^{k}\left(\frac{s\,M}{c(s+1)}\right)\\ &\leq -\frac{c\,k\,(k+1)}{4}\mathbb{E}[\|x_{k+1}-x_*\|^2] + \sum_{s=1}^{k}\left(\frac{sM}{c(s+1)}\right)\\ &\leq \frac{k}{c}M.\end{aligned}$$

If we divide both sides by $\sum_{s=1}^{k} s$, we obtain

$$\frac{\sum_{s=1}^{k} s\,\mathbb{E}[f(x_s)] - \sum_{s=1}^{k} s\,\mathbb{E}[f(x_*)]}{\sum_{s=1}^{k} s} \leq \frac{k\,M}{c\sum_{s=1}^{k} s} = \frac{2\,M}{c(k+1)}. \tag{3.22}$$

5.1. Our practical BSG method (BSG-1). In the numerical experiments for unconstrained LL BLPs ($Y(x) = \mathbb{R}^m$), we are mainly interested in testing the practical version of the BSG method (Algorithm 1), with the BSG approximation introduced in Section 1.3. This version is referred to as the BSG-1 method and requires no second-order derivatives (not even finite-difference approximations) and no Hessian-vector products; see Algorithm 2. It essentially consists of using first-order rank-1 approximations for the Hessian matrices appearing in the adjoint formula (1.2). The BSG-1 formula for the deterministic case was already given in (1.4). In the stochastic setting, we have

$$\begin{aligned}&d(x_k,\tilde{y}_k,\xi_k) \;=\\ &-\left(g_x^u(x_k,\tilde{y}_k,w_k^u) - \frac{(g_y^\ell(x_k,\tilde{y}_k,w_k^\ell))^\top g_y^u(x_k,\tilde{y}_k,w_k^u)}{(g_y^\ell(x_k,\tilde{y}_k,w_k^\ell))^\top g_y^\ell(x_k,\tilde{y}_k,w_k^\ell)} g_x^\ell(x_k,\tilde{y}_k,w_k^\ell)\right).\end{aligned} \tag{5.1}$$

Algorithm 2 BSG-1 Method (unconstrained UL case)

Input: $(x_0,y_0)\in\mathbb{R}^n\times\mathbb{R}^m$, $\{\alpha_k\}_{k\geq 0} > 0$.

For $k = 0,1,2,\ldots$ **do**

Step 1. Obtain an approximation $\tilde{y}_k$ to the LL optimal solution $y(x_k)$.

Step 2. Draw the stochastic gradients $g_x^u(x_k,\tilde{y}_k,w_k^u)$, $g_y^u(x_k,\tilde{y}_k,w_k^u)$, $g_x^\ell(x_k,\tilde{y}_k,w_k^\ell)$, and $g_y^\ell(x_k,\tilde{y}_k,w_k^\ell)$. Compute $d(x_k,\tilde{y}_k,\xi_k)$ using (5.1).

Step 3. Compute $x_{k+1} = x_k + \alpha_k\, d(x_k,\tilde{y}_k,\xi_k)$.

End do

We will also test the BSG method with stochastic Hessians, where the direction is calculated from (2.1). This version is referred to as BSG-H. The adjoint system $H_{yy}^\ell(x_k,\tilde{y}_k,w_k^\ell)\lambda = -g_y^u(x_k,\tilde{y}_k,w_k^u)$ is solved by the linear conjugate gradient method until non-positive curvature is detected.

In both BSG-1 and BSG-H versions of the BSG method, we will apply the SG method (3.25) in Step 1 of Algorithm 1 for a certain budget i_k of iterations, obtaining an approximation $\tilde{y}_k$ to the LL optimal solution $y(x_k)$. We will consider two inexact schemes for the solution of the LL problem. The first one (1 step) consists of taking a single step of SG ($i_k = 1, \forall k$). In the second one (inc. acc.), the number of SG steps increases by 1 every time the

stage, the goal of the UL and LL problems is to determine the values of λ_t and δ_t that ensure a small classification error on T_t and on all the previous tasks T_i, with $i < t$. To this end, the UL problem determines (λ_t, δ_t) by minimizing the prediction error on $D^t_{\text{val}} = \cup_{i \le t} D^i_{\text{v}}$, which is composed of the data sampled from the validation sets associated with the current and previous tasks. Similarly, the LL problem determines δ_t by minimizing the error on $D^t_{\text{train}} = \cup_{i \le t} D^i_{\text{tr}}$. Note that at each stage one solves a different problem since the objective functions of the UL and LL change. The formulation of the problem solved at stage t, with $t \in \{1, \dots, T\}$, can be written as follows

$$\begin{aligned} &\min_{(\lambda_t, \delta_t)} \frac{1}{|D^t_{\text{val}}|} \sum_{(z,u) \in D^t_{\text{val}}} \ell(\phi(z; \lambda_t, \delta_t), u) \\ &\text{s.t. } \delta_t \in \operatorname*{argmin}_{\delta_t} \frac{1}{|D^t_{\text{train}}|} \sum_{(z,u) \in D^t_{\text{train}}} \ell(\phi(z; \lambda_t, \delta_t), u). \end{aligned} \tag{4.1}$$

Moving from one task to a sequence of tasks introduces new challenges relative to the classic empirical risk minimization. In particular, once a new task is included in the problem, the performance on the previous tasks tends to deteriorate, thus resulting in the well-studied phenomenon of *catastrophic forgetting* [18]. Moreover, parameters learned on previous tasks are expected to be close to the parameters required by the new task (this is referred to as a positive transfer of knowledge between tasks), which leads to accelerating the learning process as compared with training from scratch. Solving the sequence of problems given in (4.1) allows the learning process to both overcome the catastrophic forgetting issue and take advantage of the parameters determined on the previous tasks. However, we point out that the large dimension of the datasets usually considered in ML may prevent the use of the whole sets D^i_{tr} and D^i_{v} from previous tasks i's, where $i < t$ and t is the current task. In such cases, it may be necessary to resort to subsets $\bar{D}^i_{\text{tr}} \subset D^i_{\text{tr}}$ and $\bar{D}^i_{\text{v}} \subset D^i_{\text{v}}$, which we will not do in this paper given that our interest focuses on the solution of stochastic BLPs.

5. Numerical experiments

All tests were run using MATLAB R2021a on an Acer Aspire with 8GB of RAM and an Intel Core i7-8550U processor running at 1.80GHz. In all figures, we plot an accurate approximation for the true function f of the BLP ($f(x) = f_u(x, y(x))$).

A lower bound for the left-hand side is given by

$$\min_{s=1,\dots,k} \mathbb{E}[f(x_s)] - \mathbb{E}[f(x_*)] \;\le\; \sum_{s=1}^{k} \frac{s}{\sum_{s=1}^{k} s} \mathbb{E}[f(x_s)] - \sum_{s=1}^{k} \frac{s}{\sum_{s=1}^{k} s} \mathbb{E}[f(x_*)]. \tag{3.23}$$

By combining (3.22) and (3.23), we conclude the proof. ∎

3.4. Rate in the convex case. In this subsection, we state the convergence rate of the BSG method assuming that f is convex and attains a minimizer x_*.

Assumption 3.9 (Convexity of the UL objective function)**.** *Given $(\bar{y}, y) \in \mathbb{R}^m \times \mathbb{R}^m$, the (continuously differentiable) function f is convex in x, namely,*

$$f(\bar{x}) \;\ge\; f(x) + \nabla f(x)^\top (\bar{x} - x) \text{ for all } (\bar{x}, x) \in \mathbb{R}^n \times \mathbb{R}^n. \tag{3.24}$$

Moreover, f attains a minimizer.

The next theorem states that the BSG method exhibits a sublinear convergence rate of $1/\sqrt{k}$, which implies that the convergence is slower than in the strongly convex case [27, Theorem 5.3].

Theorem 3.3. *Let Assumptions 3.1–3.7 and 3.9 hold. Consider the schema given by Algorithm 1 and assume a decaying step size of the form $\alpha_k = \bar{\alpha}/\sqrt{k}$, with $\bar{\alpha} > 0$. Given a minimizer x_* of f, the sequence of iterates yielded by Algorithm 1 satisfies*

$$\min_{s=1,\dots,k} \mathbb{E}[f(x_s)] - \mathbb{E}[f(x_*)] \;\le\; \frac{\frac{\Theta^2}{2\bar{\alpha}} + \bar{\alpha}(G_d + 2(B + C_d)\Theta)}{\sqrt{k}}.$$

3.5. Imposing a bound on the distance from the LL optimal solution. In this subsection, we want to discuss a way to enforce Assumption 3.4 when using the stochastic gradient (SG) method to solve the LL problem at x_k. Given an initial point $\tilde{y}^0_k$ and a sequence of stepsizes $\{\beta_i\}$, such a SG method can be described as

$$\tilde{y}^{i+1}_k \;=\; \tilde{y}^i_k - \beta_i g^\ell_y(x_k, \tilde{y}^i_k, w^\ell_{k,i}), \quad i = 1, \dots, i_k. \tag{3.25}$$

We start by introducing the sampling assumptions that are standard in the literature related to the SG method.

Assumption 3.10. *The stochastic gradient $g^\ell_y(x_k, \tilde{y}_k, w^\ell_k)$ is unbiased*

$$\mathbb{E}_{w^\ell_k}[g^\ell_y(x_k, \tilde{y}_k, w^\ell_k)] \;=\; \nabla_y f_\ell(x_k, \tilde{y}_k). \tag{3.26}$$

Moreover, there exists a positive constant $Q > 0$ such that

$$\mathbb{E}_{w_k^\ell}[\|g_y^\ell(x_k, \tilde{y}_k, w_k^\ell)\|^2] \;\leq\; Q^2.$$

Moreover, we assume f_ℓ to be strongly convex (Assumption 3.11 below).

Assumption 3.11 (Strong convexity of the LL objective function)**.** *Given $x \in \mathbb{R}^n$, the function f_ℓ is strongly convex in y, namely, there exists a constant $\mu > 0$ such that*

$$f_\ell(x, \bar{y}) \;\geq\; f_\ell(x, y) + \nabla f_\ell(x, y)^\top (x, \bar{y} - y) + \frac{\mu}{2}\|x, \bar{y} - y\|^2 \text{ for all } (y, \bar{y}) \in \mathbb{R}^m \times \mathbb{R}^m.$$

Assumption 3.11 implies that $f_\ell(x_k, \cdot)$ has a unique minimizer $y(x_k)$.

Recalling that the convergence rate of the SG method with decaying step-size is $\mathcal{O}(1/\sqrt{i})$, and by choosing i_k equal to k^2, one guarantees the existence of a positive constant C_y such that (3.4) holds. In fact, by choosing a decaying step size sequence $\{\beta_i\}$ given by $\beta_i = \gamma/i$, where $\gamma \geq 1/(2\mu)$ is a positive constant, and under Assumptions 3.10–3.11, from [33, Equation (2.9)] it follows that the choice $i_k = k^2$ implies (with $\tilde{y}_k = \tilde{y}_k^{i_k+1}$)

$$\mathbb{E}[\|\tilde{y}_k - y(x_k)\|^2] \;\leq\; \frac{\max\{\gamma^2 Q(2\mu\gamma - 1)^{-1}, \|\tilde{y}_k^0 - y(x_k)\|^2\}}{k^2}.$$

We point out that in order for the previous result to hold in the constrained LL case, $Y(x_k)$ is required to be a nonempty bounded closed convex set.

3.6. Imposing a bound on dynamic sampling. In this subsection, we want to mention a dynamic sampling strategy to enforce the inequality in Assumption 3.5. For the sake of simplicity, we will omit the subscript k in this subsection. Such a dynamic sampling strategy allows reducing the level of noise by increasing the size of the batch. Since Assumption 3.10 implies that $D(x, y, \xi)$ (see (2.3) for the unconstrained LL case) is an unbiased estimate of the corresponding true gradients and Hessians, let us assume that $D(x, y, \xi)$ is normally distributed with mean $D(x, y)$ (see (2.2) for the unconstrained LL case) and variance σ^2 (and that the dimension of the covariance matrix is q).

To increase the accuracy of $D(x, y, \xi)$ as an estimator of $D(x, y)$, we can choose a larger batch size, which is denoted by b_D. Let $\bar{D}(x, y, \xi) = (1/b_D)\sum_{r=1}^{b_D} D(x, y, \xi^r)$ be the corresponding mini-batch stochastic estimate, where $\{\xi^r\}_{r=1}^{b_D}$ are values sampled from ξ. It is known that (for details, see, for instance, [27, Section 5.3])

$$\mathbb{E}_{\xi_k}[\|D(x, y) - \bar{D}(x, y, \xi)\|] \;\leq\; \frac{\sigma\sqrt{q}}{\sqrt{b_D}}.$$

To guarantee that Assumption 3.5 holds, we need to choose a mini-batch size n_D and a sample standard deviation σ such that

$$\frac{\sigma\sqrt{q}}{\sqrt{b_D}} \;\leq\; C_D\, \alpha_k.$$

Therefore, when α_k decreases, the dynamic sampling strategy would increase b_D.

4. Continual learning

We are going to use instances of Continual Learning (CL) as practical stochastic bilevel problems to test the performance of BSG and DARTS. CL was briefly described in Section 1, and is now introduced in more detail. Let us denote a whole features/labels dataset by $\mathcal{D} = \{(z_j, u_j),\, j \in \{1, \ldots, N\}\}$, consisting of N pairs of a feature vector z_j and the corresponding true label u_j. For any data point j, the classification is deemed correct if the right label is predicted. To evaluate the loss incurred when using the prediction function $\phi(z; \theta)$, which in this section is supposed to be a Deep Neural Network (DNN), we use a loss function $\ell(\phi(z; \theta), u)$.

The goal of CL is to minimize the prediction error over a sequence of tasks that become available one at a time. Among the many different formulations proposed for CL, hierarchical objectives have been used in [36, 43]. In order to assess the performance of BSG against DARTS on this class of problems, we adapt the formulation in [36] to an incremental setting where each task is available as a subset of samples. Given $t \in \{1, \ldots, T\}$, let $\mathcal{D}_t$ be the set of samples for a new task T_t, which can be split into a training set D_{tr}^t and a validation set D_{v}^t. Moreover, let us split the parameters θ_t of the model $\phi(z; \theta_t)$ into two subvectors, λ_t and δ_t, whose roles are to give us flexibility in minimizing the classification error on the training and validation data along the sequence of tasks. According to [13], a reasonable strategy is to choose λ_t and δ_t as the vectors of weights in the hidden and output layers, respectively.

To solve the overall CL problem, one starts from the first task T_1 and, after an arbitrary number of iterations, we include in the problem the second task T_2. One reiterates this procedure until all the tasks have been added to the problem. Let us now suppose that one has already added t tasks. At this