humancompatible.train: Implementing Optimization Algorithms for Stochastically-Constrained Stochastic Optimization Problems

Andrii Kliachkin, Jana Lepšová, Gilles Bareilles, Jakub Marecek
Artificial Intelligence Center,
Czech Technical University in Prague
firstname.lastname@fel.cvut.cz

Abstract

There has been a considerable interest in constrained training of deep neural networks (DNNs) recently for applications such as fairness and safety. Several toolkits have been proposed for this task, yet there is still no industry standard. We present humancompatible.train (https://github.com/humancompatible/train), an easily-extendable PyTorch-based Python package for training DNNs with stochastic constraints. We implement multiple previously unimplemented algorithms for stochastically constrained stochastic optimization. We demonstrate the toolkit use by comparing two algorithms on a deep learning task with fairness constraints.

1 Introduction

There has been a considerable interest in constrained training of deep neural networks (DNNs), recently [23]. Therein, one considers an empirical risk minimization (ERM) problem with constraints involving expectations:

$$\min_{x \in \mathbb{R}^n} \mathbb{E}[f(x,\xi)] \quad \text{s.t.} \quad \mathbb{E}[c(x,\zeta)] \le 0, \tag{1}$$

where ξ and ζ are random variables, and $f(\cdot,\xi):\mathbb{R}^n\to\mathbb{R}$ and $c(\cdot,\zeta):\mathbb{R}^n\to\mathbb{R}^m$ are continuous nonsmooth nonconvex mappings. This can be seen as a special case of stochastically constrained stochastic optimization problem, whose applications in operations research and statistics go well beyond DNNs.

In this work, we present humancompatible.train: a Python toolkit for training of neural networks with constraints as in (1), and demonstrate it on a fairness use-case. This draws on numerous recent proposals [12, 1, 6, 22, 2, 20, 21, 4, 7, 24, 11, 15, 16] to solve convex and non-convex empirical-risk minimization problems subject to constraints bounding the absolute value of empirical risk. See Table 1 for an overview. Numerous other algorithms of this kind could be construed, based on a number of design choices, including: (a) sampling techniques for the ERM objective and the constraints, either the same or different; (b) use of first-order or higher-order derivatives; (c) use of globalization strategies such as filters or line search; (d) use of "true" globalization strategies including random initial points and random restarts in order to reach global minimizers.

Nevertheless, there is no established toolkit, yet, which would implement all of these algorithms and which would allow for their fair comparison. This is not to say that there are no toolkits developed in this direction. Indeed, the package CHOP [14] implements Proximal Gradient Descent [19], Frank-Wolfe [18], and the Stochastic Three-Composite Minimization [26, 25], but is not actively maintained anymore. These algorithms require the projection, which is non-trivial in training general

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Constrained Optimization for Machine Learning (COML).

^{*}Equal contribution, more junior authors listed first.

DNNs, to be efficiently implementable. There is the recently proposed Cooper [13], that focuses on translating constrained problem to the unconstrained problem of minimizing the augmented Lagrangian. In contrast with unconstrained minimization of stochastic nonconvex objectives, where convergence guarantees exists [9], no convergence guarantees are available for the Augmented Lagrangian method in the stochastic nonconvex objective and constraints setting; see SSL-ALM [15]. There is also GeoTorch [17], which focuses on particular smooth manifolds encountered, e.g., in quantum information theory. While our toolkit is still work in progress, we implement several algorithms that have never been implemented before, such as SSL-ALM [15], Stochastic Switching Subgradient [16] and the Stochastic Ghost [11].

Our contributions. The contributions of this paper are:

- a literature review of algorithms subject to handling (1);
- a toolbox that implements algorithms applicable in constrained training of DNNs;
- numerical experiments that compare these algorithms on a real-world dataset.

2 Algorithms

Solving (1) encounters the following challenges:

- large-scale objective and constraint functions, which require sampling schemes,
- the necessity of incorporating inequality constraints, not merely equality constraints,
- the necessity to cope with the nonconvexity and nonsmoothness, due to the nature of neural networks.

In this section, we identify the algorithms that address these challenges most precisely. However, we note that there exists currently no algorithm with guarantees for such a general setting.

Notation. We denote the projection of a point x onto a set \mathcal{X} by $\operatorname{proj}_{\mathcal{X}}(x) = \arg\min_{v \in \mathcal{X}} \|x - v\|^2$. We distinguish between the random variable ξ associated with the objective function and the random variable ζ associated with the constraint function. We denote \mathcal{P}_{ξ} and \mathcal{P}_{ζ} their probability distributions.

2.1 Review of methods for constrained ERM

We compare recent constrained optimization algorithms considering a stochastic objective function in Table 1. We note that most of them do not consider the case of stochastic constraints. Among those which do consider stochastic constraints, only three admit inequality constraints. Moreover, with the exception of [16], all the algorithms in Table 1 assume F to be at least C^1 , which makes addressing the challenge of nonsmoothness of F infeasible. The recent paper [8] leads us to the conclusion that assuming the objective and constraint functions to be tame and locally Lipschitz is a suitable requirement for solving (1) with theoretical guarantees of convergence. At this point, however, no such algorithm exists, to the best of our knowledge. Consequently, we consider the practical performance of the algorithms that address the challenges of solving (1) most closely: SSL-ALM [15], and Stoch. Switching Subgradient [16].

2.2 Stochastic Smoothed and Linearized AL Method (SSL-ALM)

The Stochastic Smoothed and Linearized AL Method (SSL-ALM) was described in [15] for optimization problems with stochastic linear constraints. Although problem (1) has non-linear inequality constraints, we use the SSL-ALM due to the lack of algorithms in the literature dealing with stochastic non-linear constraints; see Table 1. The transition between equality and inequality constraints can be handled by either taking the maximum between the constraint value and 0 or using slack variables. Following the structure of [15], we minimize over the set $\mathcal{X} = \mathbb{R}^n \times \mathbb{R}^m_{\geq 0}$. The method is based on the augmented Lagrangian (AL) function $L_\rho(x,y) = F(x) + y^\top C(x) + \frac{\rho}{2} \|C(x)\|^2$, which is a result of merging the Lagrange function with the penalty methods [3]. Adding a smoothing term and a variable $z \in \mathbb{R}^n$ yields the proximal AL function

$$K_{\rho,\mu}(x,y,z) = L_{\rho}(x,y) + \frac{\mu}{2} ||x-z||^2.$$

The SSL-ALM method was originally proposed in [15] where it is interpreted as an inexact gradient descent step on the Moreau envelope. An important property of the Moreau envelope is that its stationary points coincide with those of the original function.

In each iteration, we sample $\xi \stackrel{iid}{\sim} \mathcal{P}_{\xi}$ to evaluate the objective and ζ_1 , $\zeta_2 \stackrel{iid}{\sim} \mathcal{P}_{\zeta}$ to evaluate the constraint function and its Jacobian matrix, respectively. The function

$$G(x, y, z; \xi, \zeta_1, \zeta_2) = \nabla f(x, \xi) + \nabla c(x, \zeta_1)^{\mathsf{T}} y + \rho \nabla c(x, \zeta_1)^{\mathsf{T}} c(x, \zeta_2) + \mu(x - z) \tag{2}$$

is defined so that, in iteration k, $\mathbb{E}_{\xi,\zeta_1,\zeta_2}[G(x_k,y_{k+1},z_k;\xi,\zeta_1,\zeta_2)] = \nabla K_{\rho,\mu}(x_k,y_{k+1},z_k)$. Omitting some details, the updates are performed using some parameters η,τ , and β as follows:

$$y_{k+1} = y_k + \eta c(x, \zeta_1),$$

$$x_{k+1} = \text{proj}_{\mathcal{X}}(x_k - \tau G(x_k, y_{k+1}, z_k; \xi, \zeta_1, \zeta_2)),$$

$$z_{k+1} = z_k + \beta(x_k - z_k).$$
(3)

2.3 Stochastic Switching Subgradient Method (SSw)

The Stochastic Switching Subgradient method was described in [16] for optimization problems over a closed convex set $\mathcal{X} \subset \mathbb{R}^d$ which is easy to project on and for weakly convex objective and constraint functions F and C which may be non-smooth. This is why the notion of gradient of F and C is replaced by a more general notion of subgradient, which is an element of a subdifferential.

The algorithm requires as input a prescribed sequence of infeasibility tolerances ϵ_k and sequences of stepsizes η_k^f and η_k^c . In iteration k, we sample $\zeta_1,\ldots,\zeta_J\stackrel{iid}{\sim}\mathcal{P}_\zeta$ to compute an estimate $\overline{c}^J(x_k)$. If $\overline{c}^J(x_k)$ is smaller than ϵ_k , we sample $\xi\stackrel{iid}{\sim}\mathcal{P}_\xi$ and an update between x_k and x_{k+1} is computed using a stochastic estimate $S^f(x_k,\xi)$ of an element of the subdifferential $\partial F(x_k)$ of the objective function:

$$x_{k+1} = \operatorname{proj}_{\mathcal{X}}(x_k - \eta_k^f S^f(x_k, \xi)).$$

Otherwise, we sample $\zeta \stackrel{iid}{\sim} \mathcal{P}_{\zeta}$ and the update is computed using a stochastic estimate $S^c(x_k,\zeta)$ of an element of the subdifferential $\partial C(x_k)$ of the constraint function:

$$x_{k+1} = \operatorname{proj}_{\mathcal{X}}(x_k - \eta_k^c S^c(x_k, \zeta)).$$

The algorithm presented here is slightly more general than the one presented in [16]: we allow for the possibility of different stepsizes for the objective update, η_k^f , and the constraint update η_k^c , while the original method employs equal stepsizes $\eta_k^f = \eta_k^c$.

2.4 Package implementation

In the package, both algorithms are implemented with an API close to that of a PyTorch Optimizer. In addition to the step() method, which performs the primal update, we add a dual_step() method, which updates the dual parameters and performs other related tasks. This allows for pipelines similar to those of PyTorch.

3 Experimental evaluation

Data. In this section, we illustrate the presented algorithms on a real-world instance of the ACS dataset, which is based on the ACS PUMS data sample (American Community Survey Public Use Microdata Sample), accessed in Python via the Folktables package [10]. In particular, we use the ACSIncome dataset over the state of Virginia, and choose the binary classification task of predicting whether an individual's income is over \$50,000. The dataset contains 9 features and 46,144 data points. We choose marital status (**MAR**) as the protected attribute, which is a categorical attribute with 5 values: Married, Widowed, Divorced, Separated, and Never Married/Under 15. The dataset is split randomly into train (80%, 36,915 points) and test (20%, 9,229 points) subsets. The protected attribute is then removed from the data so that the model cannot learn from it directly. The data is normalized using Scikit-Learn StandardScaler.

Using a utility provided with the humancompatible.train package, we sample equal number of data points from each subgroup at each iteration to ensure that all constraints can be computed.

Table 1: Assumptions on objective and constraint functions, F and C , which allow for theoretical
convergence proofs.

		Objective function F				Constraint function C						
Algorithm	stochastic	weakly convex	\mathcal{C}^1 with Lipschitz $ abla F$	tame loc. Lipschitz	stochastic	C(x) = 0	$C(x) = 0$ and $C(x) \le 0$	linear	weakly convex	\mathcal{C}^1 with Lipschitz $ abla C$	tame loc. Lipschitz	
SGD	/	(/)	(√)	✓								
[2] [12] [6]	/	_	/	_	-	/	_	-	_	/	_	
[20]	1	-	$\checkmark(\mathcal{C}^3)$	-	-	✓	_	-	-	$\checkmark(\mathcal{C}^3)$	-	
[24] [7]	/	_	✓	-	_	(\checkmark)	✓	-	-	✓	-	
[21]	1	_	$\checkmark(\mathcal{C}^2)$	-	-	(\checkmark)	✓	-	-	$\mathcal{I}(\mathcal{C}^2)$	-	
[4]	/	_	√ (+ cvx)	-	_	/	_	/	_	_	_	
[22]	/	_	1	-	/	1	-	-	_	✓	_	
SSL-ALM [15]	1	_	/	_	/	(/)	/	/	_	-	-	
Stoch. Ghost [11]	✓.	_	1	_	/	(/)	✓.	-	-	✓	-	
Stoch. Switch. Subg. [16]	/	/	_	-	/	(\checkmark)	✓	-	/	-	-	

Problems. In all problems, we set the objective function the Binary Cross Entropy with Logits Loss

$$\ell(f_{\theta}(X_i), Y_i) = -Y_i \cdot \log \sigma(f_{\theta}(X_i)) - (1 - Y_i) \cdot \log(1 - \sigma(f_{\theta}(X_i))), \tag{4}$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the sigmoid function, and the prediction function f_{θ} is a neural network with 2 interconnected hidden layers of sizes 64 and 32 and ReLU activation, with a total of 194 parameters.

For the constrained algorithms, for each of the 5 groups (defined by the value of the protected attribute), we put an upper bound on the absolute difference between Positive Rate computed on that group and the overall Positive Rate:

$$|P(f_{\theta}(X) = 1 \mid G = g) - P(f_{\theta}(X) = 1)| \le c,$$
 (5)

where G is the group membership indicator.

We set the constraint bound c = 0.05 for all groups; we use Fairret [5] to compute the constraints.

Algorithms and parameters. We compare the performance of two algorithms for solving the constrained problem: (1) SSL-ALM (Sec. 2.2 - parameters $\mu=2.0$, $\rho=1.0$, $\tau=0.05$, $\eta=0.1$, $\beta=0.5$, $M_y=100$), and (2) Stochastic Switching Subgradient (SSw) (Sec. 2.3 - $\eta_k^f=0.01$, $\eta_k^c=0.01$, $\epsilon_1=10^{-1}$, $\epsilon_k=\frac{\epsilon_{k-1}}{\sqrt{k}}$ for all $k\geq 1$). As a baseline, we also train the network without constraints using Adam with default hyperparameter values found in PyTorch.

Setup. Experiments are conducted on an Asus Zenbook UX535 laptop with AMD Ryzen 7 5800H CPU, and 16GB RAM. Each algorithm is run for 1 minute, repeated 5 times.

Optimization performance. Figure 1 presents the evolution of loss and constraint values over the train and test datasets for the baseline (Adam) (row 1), and the two algorithms addressing the constrained problem (SSL-ALM on row 2 and SSw on row 3). Both SSL-ALM and SSw algorithms succeed in keeping the constraint values within bounds, with SSL-ALM minimizing the objective function faster under the chosen hyperparameters.

Acknowledgements

The work was funded by European Union's Horizon Europe research and innovation program under grant agreement No. 101070568.

References

[1] Albert Berahas, Frank E. Curtis, Daniel Robinson, and Baoyu Zhou. Sequential quadratic optimization for nonlinear equality constrained stochastic optimization. *SIAM Journal on Optimization*, 31:1352–1379, 05 2021.

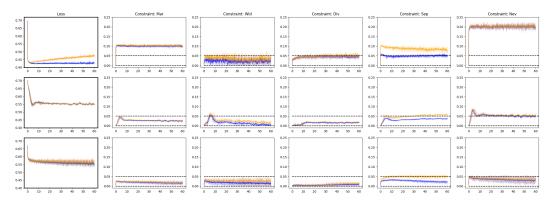


Figure 1: Train (blue) and test (orange) statistics over time (s) on the ACS Income dataset for each algorithm: Adam (top-row), SSL-ALM (middle-row), and SSw (bottom-row). The plots depict the mean values for loss (leftmost column) and constraints (second to rightmost column) at each timestamp (rounded to the nearest 0.1 seconds) over 5 runs. The shaded area depicts the region between the lowest and the highest value for corresponding statistics calculated the same way.

- [2] Albert S. Berahas, Frank E. Curtis, Michael J. O'Neill, and Daniel P. Robinson. A stochastic sequential quadratic optimization algorithm for nonlinear equality constrained optimization with rank-deficient jacobians, 2023.
- [3] D.P. Bertsekas and W. Rheinboldt. *Constrained Optimization and Lagrange Multiplier Methods*. Computer science and applied mathematics. Academic Press, 2014.
- [4] Raghu Bollapragada, Cem Karamanli, Brendan Keith, Boyan Lazarov, Socratis Petrides, and Jingyi Wang. An adaptive sampling augmented lagrangian method for stochastic optimization with deterministic constraints. *Computers and Mathematics with Applications*, 149:239–258, 2023.
- [5] Maarten Buyl, Marybeth Defrance, and Tijl De Bie. fairret: a framework for differentiable fairness regularization terms. In *International Conference on Learning Representations*, 2024.
- [6] Frank E. Curtis, Michael J. O'Neill, and Daniel P. Robinson. Worst-case complexity of an sqp method for nonlinear equality constrained stochastic optimization. *Mathematical Programming*, 205(1):431–483, May 2024.
- [7] Frank E. Curtis, Daniel P. Robinson, and Baoyu Zhou. Sequential quadratic optimization for stochastic optimization with deterministic nonlinear inequality and equality constraints. *SIAM Journal on Optimization*, 34(4):3592–3622, 2024.
- [8] Damek Davis, Dmitriy Drusvyatskiy, Sham Kakade, and Jason D. Lee. Stochastic subgradient method converges on tame functions, 2018.
- [9] Damek Davis, Dmitriy Drusvyatskiy, Sham Kakade, and Jason D. Lee. Stochastic Subgradient Method Converges on Tame Functions. *Foundations of Computational Mathematics*, 20(1):119– 154, February 2020.
- [10] Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. Retiring adult: New datasets for fair machine learning. *Advances in Neural Information Processing Systems*, 34, 2021.
- [11] Francisco Facchinei and Vyacheslav Kungurtsev. Stochastic approximation for expectation objective and expectation inequality-constrained nonconvex optimization, 2023.
- [12] Yuchen Fang, Sen Na, Michael W. Mahoney, and Mladen Kolar. Fully stochastic trust-region sequential quadratic programming for equality-constrained optimization problems. *SIAM Journal on Optimization*, 34(2):2007–2037, 2024.
- [13] Jose Gallego-Posada, Juan Ramirez, Meraj Hashemizadeh, and Simon Lacoste-Julien. Cooper: A Library for Constrained Optimization in Deep Learning. *arXiv preprint arXiv:2504.01212*, 2025.

- [14] Fabian Pedregosa Geoffrey Negiar. Chop: continuous optimization built on pytorch. 2020.
- [15] Ruichuan Huang, Jiawei Zhang, and Ahmet Alacaoglu. Stochastic smoothed primal-dual algorithms for nonconvex optimization with linear inequality constraints, 2025.
- [16] Yankun Huang and Qihang Lin. Oracle complexity of single-loop switching subgradient methods for non-smooth weakly convex functional constrained optimization, 2023.
- [17] Mario Lezcano-Casado. Trivializations for gradient-based optimization on manifolds. In *Advances in Neural Information Processing Systems, NeurIPS*, pages 9154–9164, 2019.
- [18] Francesco Locatello, Alp Yurtsever, Olivier Fercoq, and Volkan Cevher. Stochastic frank-wolfe for composite convex minimization. Advances in Neural Information Processing Systems, 32, 2019.
- [19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.
- [20] Sen Na, Mihai Anitescu, and Mladen Kolar. An adaptive stochastic sequential quadratic programming with differentiable exact augmented lagrangians. *Mathematical Programming*, 199(1):721–791, May 2023.
- [21] Sen Na, Mihai Anitescu, and Mladen Kolar. Inequality constrained stochastic nonlinear optimization via active-set sequential quadratic programming, 2023.
- [22] Figen Oztoprak, Richard Byrd, and Jorge Nocedal. Constrained optimization in the presence of noise. *SIAM Journal on Optimization*, 33(3):2118–2136, 2023.
- [23] Juan Ramirez, Meraj Hashemizadeh, and Simon Lacoste-Julien. Position: Adopt constraints over penalties in deep learning, 2025.
- [24] Qiankun Shi, Xiao Wang, and Hao Wang. A momentum-based linearized augmented lagrangian method for nonconvex constrained stochastic optimization. *Optimization Online*, 2022.
- [25] Alp Yurtsever, Varun Mangalick, and Suvrit Sra. Three operator splitting with a nonconvex loss function. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12267–12277. PMLR, 18–24 Jul 2021.
- [26] Alp Yurtsever, Bang Công Vu, and Volkan Cevher. Stochastic three-composite convex minimization. Advances in Neural Information Processing Systems, 29, 2016.