Journal of Machine Learning Research ☺ (2022) ☺ Submitted 11/21; Published ☺/☺

NCVX: A User-Friendly and Scalable Package for Nonconvex Optimization in Machine Learning

Buyun Liang LIANG664@UMN.EDU
Ju Sun JUSUN@UMN.EDU
Department of Computer Science & Engineering
University of Minnesota, Twin Cities
Minneapolis, MN 55455, USA

Editor: ☺ and ☺

Abstract

Optimizing nonconvex (NCVX) problems, especially those nonsmooth (NSMT) and constrained (CSTR), is an essential part of machine learning and deep learning. But it is hard to reliably solve this type of problems without optimization expertise. Existing general-purpose NCVX optimization packages are powerful, but typically cannot handle nonsmoothness. `GRANSO` is among the first packages targeting NCVX, NSMT, CSTR problems. However, it has several limitations such as the lack of auto-differentiation and GPU acceleration, which preclude the potential broad deployment by non-experts. To lower the technical barrier for the machine learning community, we revamp `GRANSO` into a user-friendly and scalable python package named `NCVX`, featuring auto-differentiation, GPU acceleration, tensor input, scalable QP solver, and zero dependency on proprietary packages. As a highlight, `NCVX` can solve general CSTR deep learning problems, the first of its kind. `NCVX` is available at `https://ncvx.org`, with detailed documentation and numerous examples from machine learning and other fields.

Keywords: BFGS-SQP, second-order methods, nonconvex optimization, nonsmooth optimization, constrained optimization, auto-differentiation, GPU acceleration, PyTorch

1. Introduction

Mathematical optimization is an indispensable modeling and computational tool for all science and engineering fields, especially for machine and deep learning. To date, researchers have developed numerous foolproof techniques and user-friendly solvers and modeling languages for convex (CVX) problems, such as SDPT3 (Toh et al., 1999), Gurobi (Gurobi Optimization, LLC, 2021), Cplex (Cplex, 2009), TFOCS (Becker et al., 2011), CVX(PY) (Grant et al., 2008; Diamond and Boyd, 2016), AMPL (Gay, 2015), YALMIP (Lofberg, 2004). These developments have substantially lowered the barrier of CVX optimization for non-experts. However, practical problems, especially from machine and deep learning, are often nonconvex (NCVX), and possibly also nonsmooth (NSMT) and constrained (CSTR).

There are methods and packages handling NCVX problems in restricted settings: PyTorch (Paszke et al., 2019) and TensorFlow (Abadi et al., 2015) can solve large-scale NCVX, NSMT problems without constraints. CSTR problems can be heuristically turned into penalty forms and solved as unconstrained, but this may not produce feasible solutions for the original problems. When the constraints are simple, structured methods such as projected (sub)gradient and Frank-Wolfe (Sra et al., 2012) can be used. When the constraints are differentiable manifolds, one can consider manifold optimization methods and packages, e.g., (Py)manopt (Boumal et al., 2014; Townsend et al., 2016) and Geomstats (Miolane et al., 2020). For general CSTR problems, KNITRO (Pillo and Roma, 2006) and IPOPT (Wächter and Biegler, 2005) implement interior-point methods, while ensmallen (Curtin et al., 2021) and GENO (Laue et al., 2019) rely on augmented Lagrangian methods. However, beyond smooth (SMT) constraints both families of methods can at best handle special NSMT constraints. Packages specialized for machine learning, such as scikit-learn (Pedregosa et al., 2011), MLib (Meng et al., 2016) and Weka (Witten et al., 2005), often use problem-specific solvers that cannot be easily extended to new formulations.

2. The `GRANSO` and `NCVX` packages

`GRANSO`[1] is among the first optimization packages that can handle general NCVX, NSMT, CSTR problems (Curtis et al., 2017):

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x}), \;\; \text{s.t. } c_i(\boldsymbol{x}) \leq 0, \; \forall \, i \in \mathcal{I}; \;\; c_i(\boldsymbol{x}) = 0, \; \forall \, i \in \mathcal{E}. \tag{1}$$

Here, the objective f and constraint functions c_i's are only required to be almost everywhere continuously differentiable. `GRANSO` is based on quasi-Newton methods with sequential quadratic programming (BFGS-SQP), and has the following advantages: (1) **unified treatment of NCVX problems**: no need to distinguish CVX vs NCVX and SMT vs NSMT problems, similar to typical nonlinear programming packages; (2) **reliable step size rule**: specialized methods for NSMT problems, such as subgradient and proximal methods, often entail tricky step size tuning and require the expertise to recognize the structures (Sra et al., 2012). By contrast, `GRANSO` chooses step sizes adaptively via gold standard line search; (3) **principled stopping criterion**: `GRANSO` stops its iteration by checking a theory-grounded stationarity condition for NMST problems, whereas specialized methods are usually stopped when reaching ad-hoc iteration caps.

However, `GRANSO` suffers from several practicality limitations. To overcome these and facilitate practical usage in machine and deep learning, we revamp `GRANSO` with crucial enhancements and turn it into our `NCVX` package, which is based on PyTorch. The limitations and our corresponding enhancements include: (1) `GRANSO` requires analytical subgradients, whereas `NCVX` removes this need and performs auto-differentiation; (2) `GRANSO` only supports CPU-based computation, whereas `NCVX` both CPUs and GPUs to allow massively-parallel computation; (3) `GRANSO` defaults variables as vectors, while `NCVX` allows general tensor variables including vectors and matrices; (4) `GRANSO` solves two QP instances per iteration and uses MATLAB's QP solver that hardly scales up. `NCVX` integrates OSQP (Stellato et al., 2020) that outperforms commercial QP solvers in terms of scalability and speed; (5) `GRANSO` is written in MATLAB, which is proprietary software. All the dependencies of `NCVX` are open-source and non-proprietary. All these enhancements are crucial for machine learning researchers and practitioners to solve large-scale problems.

`NCVX` is available at the GitHub repository `https://github.com/sun-umn/NCVX` under the MIT license, along with a documentation website `https://ncvx.org` which includes nu-

1. `http://www.timmitchell.com/software/GRANSO/`

merous detailed tutorials. For potential contributors and collaborators, our GitHub repository is a convenient place to report issues, seek help, and make code contributions.

3. Usage Examples: Dictionary Learning and Neural Perceptual Attack

In order to make `NCVX` friendly to non-experts, we strive to keep the user input minimal. The user is only required to specify the optimization variables (names and dimensions of variables) and define the objective and constraint functions. If GPU computation is desired, the user can easily set the `device` argument. Here, we briefly demonstrate the usage of `NCVX` on orthogonal dictionary learning (Bai et al., 2018) and neural perceptual attack (Laidlaw et al., 2020), representing classical machine learning and modern deep learning, respectively.

Orthogonal Dictionary Learning (ODL) One hopes to find a "transformation" $\boldsymbol{q} \in \mathbb{R}^n$ to sparsify a data matrix $\boldsymbol{Y} \in \mathbb{R}^{n \times m}$:

$$\min_{\boldsymbol{q} \in \mathbb{R}^n} f(\boldsymbol{q}) \doteq \frac{1}{m} \left\| \boldsymbol{q}^\intercal \boldsymbol{Y} \right\|_1, \quad \text{s.t. } \left\| \boldsymbol{q} \right\|_2 = 1, \tag{2}$$

where the sphere constraint $\|\boldsymbol{q}\|_2 = 1$ is to avoid the trivial solution $\boldsymbol{q} = \boldsymbol{0}$. Problem (2) is NCVX, NSMT, and CSTR: nonsmoothness comes from the objective, and nonconvexity comes from the constraint. Demo 1 and Demo 2 show the implementations of ODL in `GRANSO` and `NCVX`, respectively. Note that the analytical gradients of the objective and constraint functions are not required in `NCVX`. Figure 1 shows that `NCVX` produces results that are faithful to that of `GRANSO` on ODL.

```
function[f,fg]=obj(q,m,Y)
   f = 1/m*norm(q'*Y, 1); % obj
   fg = 1/m*Y*sign(Y'*q); % obj grad
end
function[ce,ceg]=ce(q)
   ce = q'*q - 1; % eq constr
   ceg = 2*q; % eq constr grad
end
soln = granso(n,@(q)obj(q,m,Y),[],@ce);
```

Demo 1: `GRANSO` for ODL

```
def comb_fn(X_struct):
   q = X_struct.q
   q.requires_grad_(True) # autodiff
   f = 1/m*norm(q.T@Y, p=1) # obj
   ce = GeneralStruct()
   ce.c1 = q.T@q - 1 # eq constr
   return [f,None,ce]
var_in = {"q": [n,1]} # define variable
soln = ncvx(comb_fn, var_in)
```

Demo 2: `NCVX` for ODL

Neural Perceptual Attack (NPA) The constrained deep learning problem, NPA, is shown below:

$$\max_{\widetilde{\boldsymbol{x}}} \mathcal{L}\left(f\left(\widetilde{\boldsymbol{x}}\right), y\right), \quad \text{s.t. } d\left(\boldsymbol{x}, \widetilde{\boldsymbol{x}}\right) = \left\| \phi\left(\boldsymbol{x}\right) - \phi\left(\widetilde{\boldsymbol{x}}\right) \right\|_2 \leq \epsilon. \tag{3}$$

Here, $\boldsymbol{x}$ is an input image, and the goal is to find its perturbed version $\widetilde{\boldsymbol{x}}$ that is perceptually similar to $\boldsymbol{x}$ (encoded by the constraint) but can fool the classifier f (encoded by the objective). The loss $\mathcal{L}(\cdot, \cdot)$ is the margin loss used in Laidlaw et al. (2020). Both f in the objective and ϕ in the constraint are deep neural networks with ReLU activations, making

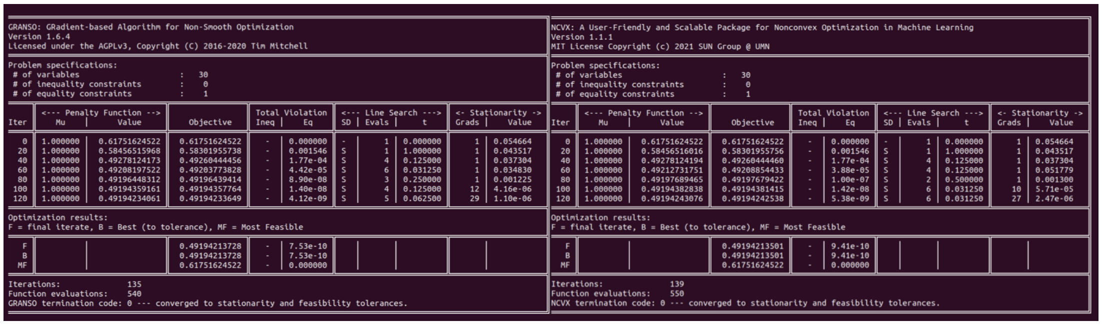

Figure 1: Consistency of `GRANSO` (left) and `NCVX` (right) on ODL

both the objective and constraint functions NSMT and NCVX. The $d(\boldsymbol{x}, \widetilde{\boldsymbol{x}})$ distance is called the Learned Perceptual Image Patch Similarity (LPIPS) (Laidlaw et al., 2020; Zhang et al., 2018). Demo 3 is the `NCVX` example for solving problem (3). Note that the data, model, loss function, and LPIPS distance definition are not included here. It is almost impossible to derive analytical subgradients for problem (3), and thus the auto-differentiation feature in `NCVX` is necessary for solving it.

```python
def comb_fn(X_struct):
    adv_inputs = X_struct.x_tilde
    adv_inputs.requires_grad_(True) # autodiff
    f = MarginLoss(model(adv_inputs),labels) # obj
    ci = GeneralStruct()
    ci.c1 = lpips_dists(adv_inputs) - 0.5 # ineq constr. percep bound epsilon=0.5
    return [f,ci,None] # No eq constr
var_in = {"x_tilde": list(inputs.shape)} # define variable
soln = ncvx(comb_fn,var_in)
```

Demo 3: `NCVX` for NPA

4. Road map

Although `NCVX` already has many powerful features, we plan to further improve it by adding several major components: (1) **symmetric rank one (SR1)**: SR1, another major type of quasi-Newton methods, allows less stringent step size search and tends to help escape from saddle points faster by taking advantage of negative curvature directions (Dauphin et al., 2014); (2) **stochastic algorithms**: in machine learning, computing with large-scale datasets often involves finite sums with huge number of terms, calling for (mini-batch) stochastic algorithms for reduced per-iteration cost and better scalability (Sun, 2019); (3) **conic programming (CP)**: semidefinite programming and second-order cone programming, special cases of CP, are abundant in machine learning, e.g., kernel machines (Zhang et al., 2019); (4) **minimax optimization (MMO)**: MMO is an emerging technique in modern machine learning, e.g., generative adversarial networks (GANs) (Goodfellow et al., 2020) and multi-agent reinforcement learning (Jin et al., 2020).

Acknowledgments

We would like to thank the `GRANSO` developers. This work was supported by UMII Seed Grant Program and NSF CMMI 2038403.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Yu Bai, Qijia Jiang, and Ju Sun. Subgradient descent learns orthogonal dictionaries. *arXiv preprint arXiv:1810.10702*, 2018.

Stephen R Becker, Emmanuel J Candès, and Michael C Grant. Templates for convex cone problems with applications to sparse signal recovery. *Mathematical programming computation*, 3(3):165, 2011.

Nicolas Boumal, Bamdev Mishra, P-A Absil, and Rodolphe Sepulchre. Manopt, a matlab toolbox for optimization on manifolds. *The Journal of Machine Learning Research*, 15 (1):1455–1459, 2014.

IBM ILOG Cplex. V12. 1: User's manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.

Ryan R Curtin, Marcus Edel, Rahul Ganesh Prabhu, Suryoday Basak, Zhihao Lou, and Conrad Sanderson. The ensmallen library for flexible numerical optimization. *Journal of Machine Learning Research*, 22(166):1–6, 2021.

Frank E Curtis, Tim Mitchell, and Michael L Overton. A bfgs-sqp method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles. *Optimization Methods and Software*, 32(1):148–181, 2017.

Yann Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *arXiv preprint arXiv:1406.2572*, 2014.

Steven Diamond and Stephen Boyd. Cvxpy: A python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research*, 17(1):2909–2913, 2016.

David M Gay. The ampl modeling language: An aid to formulating and solving optimization problems. In *Numerical analysis and optimization*, pages 95–116. Springer, 2015.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Michael Grant, Stephen Boyd, and Yinyu Ye. Cvx: Matlab software for disciplined convex programming, 2008.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021. URL `https://www.gurobi.com`.

Chi Jin, Praneeth Netrapalli, and Michael Jordan. What is local optimality in nonconvex-nonconcave minimax optimization? In *International Conference on Machine Learning*, pages 4880–4889. PMLR, 2020.

Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. Perceptual adversarial robustness: Defense against unseen threat models. *arXiv preprint arXiv:2006.12655*, 2020.

Sören Laue, Matthias Mitterreiter, and Joachim Giesen. Geno–generic optimization for classical machine learning. *arXiv preprint arXiv:1905.13587*, 2019.

J. Lofberg. YALMIP : a toolbox for modeling and optimization in MATLAB. In *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*. IEEE, 2004. doi: 10.1109/cacsd.2004.1393890.

Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241, 2016.

Nina Miolane, Nicolas Guigui, Alice Le Brigant, Johan Mathe, Benjamin Hou, Yann Thanwerdas, Stefan Heyder, Olivier Peltre, Niklas Koep, Hadi Zaatiti, et al. Geomstats: A python package for riemannian geometry in machine learning. *Journal of Machine Learning Research*, 21(223):1–9, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Gianni Pillo and Massimo Roma. *Large-scale nonlinear optimization*, volume 83. Springer Science & Business Media, 2006.

Suvrit Sra, Sebastian Nowozin, and Stephen J Wright. *Optimization for machine learning*. Mit Press, 2012.

B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4): 637–672, 2020. doi: 10.1007/s12532-020-00179-2. URL `https://doi.org/10.1007/s12532-020-00179-2`.

Ruoyu Sun. Optimization for deep learning: theory and algorithms. *arXiv preprint arXiv:1912.08957*, 2019.

Kim-Chuan Toh, Michael J Todd, and Reha H Tütüncü. Sdpt3—a matlab software package for semidefinite programming, version 1.3. *Optimization methods and software*, 11(1-4): 545–581, 1999.

James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *arXiv preprint arXiv:1603.03236*, 2016.

Ian H Witten, Eibe Frank, Mark A Hall, CJ Pal, and MINING DATA. Practical machine learning tools and techniques. In *DATA MINING*, volume 2, page 4, 2005.

Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, apr 2005. doi: 10.1007/s10107-004-0559-y.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

Richard Y Zhang, Cédric Josz, and Somayeh Sojoudi. Conic optimization for control, energy systems, and machine learning: Applications and algorithms. *Annual Reviews in Control*, 47:323–340, 2019.