Global optimization software library for research and education

Nadia Udler‡*

Abstract—Machine learning models are often represented by functions given by computer programs. Optimization of such functions is a challenging task because traditional derivative based optimization methods with guaranteed convergence properties cannot be used.. This software allows to create new optimization methods with desired properties, based on basic modules. These basic modules are designed in accordance with approach for constructing global optimization methods based on potential theory [KAP]. These methods do not use derivatives of objective function and as a result work with nondifferentiable functions (or functions given by computer programs, or black box functions), but have guaranteed convergence. The software helps to understand principles of learning algorithms. This software may be used by researchers to design their own variations or hybrids of known heuristic optimization methods. It may be used by students to understand how known heuristic optimization methods work and how certain parameters affect the behavior of the method.

Index Terms—global optimization, black-box functions, algorithmically defined functions, potential functions

Introduction

Optimization lies at the heart of machine learning and data science. One of the most relevant problems in machine learning is automatic selection of the algorithm depending on the objective. This is necessary in many applications such as robotics, simulating biological or chemical processes, trading strategies optimization, to name a few [KHNT]. We developed a library of optimization methods as a first step for self-adapting algorithms. Optimization methods in this library work with all objectives including very onerous ones, such as black box functions and functions given by computer code, and the convergences of methods is guaranteed. This library allows to create customized derivative free learning algorithms with desired properties by combining building blocks from this library or other Python libraries.

The library is intended primarily for educational purposes and its focus is on transparency of the methods rather than on efficiency of implementation.

The library can be used by researches to design optimization methods with desired properties by varying parameters of the general algorithm.

As an example, consider variant of simulated annealing (SA) proposed in [FGSB] where different values of parameters (Boltzman distribution parameters, step size, etc.) are used depending of the distance to optimal point. In this paper the basic SA algorithm is used as a starting point. We can offer more basic module as a starting point (and by specifying distribution as 'exponential' get the variant of SA) thus achieving more flexible design opportunities for custom optimization algorithm. Note that convergence of the newly created hybrid algorithm does not need to be verified when using minpy basic modules, whereas previously mentioned SA-based hybrid has to be verified separately (see [GLUQ])

Testing functions are included in the library. They represent broad range of use cases covering above mentioned difficult functions. In this paper we describe the approach underlying these optimization methods. The distinctive feature of these methods is that they are not heuristic in nature. The algorithms are derived based on potential theory [KAP], and their convergence is guaranteed by their derivation method [KPP]. Recently potential theory was applied to prove convergence of well known heuristic methods, for example see [BIS] for convergence of PSO, and to re prove convergence of well known gradient based methods, in particular, first order methods - see [NBAG] for convergence of gradient descent and [ZALO] for mirror descent. For potential functions approach for stochastic first order optimization methods see [ATFB].

Outline of the approach

The approach works for non-smooth or algorithmically defined functions. For detailed description of the approach see [KAP], [KP]. In this approach the original optimization problem is replaced with a randomized problem, allowing the use of Monte-Carlo methods for calculating integrals. This is especially important if the objective function is given by its values (no analytical formula) and derivatives are not known. The original problem is restated in the framework of gradient (sub gradient) methods, employing the standard theory (convergence theorems for gradient (sub gradient) methods), whereas no derivatives of the objective function are needed. At the same time, the method obtained is a method of nonlocal search unlike other gradient methods. It will be shown, that instead of measuring the gradient of the objective function we can measure the gradient of the potential function at each iteration step , and the value of the gradient can be obtained using values of objective function only, in the framework of Monte Carlo methods for calculating integrals. Furthermore, this value does not have to be precise, because it is recalculated at each iteration step. It will also be shown that well-known zero-order optimization methods (methods that do not use derivatives of objective function but its values only)

* *Corresponding author: nadiakap@optonline.net*
‡ *University of Connecticut (Stamford)*

are generalized into their adaptive extensions. The generalization of zero-order methods (that are heuristic in nature) is obtained using standardized methodology, namely, gradient (sub gradient) framework. We consider the unconstrained optimization problem

$$f(x_1, x_2, ..x_n) \to \min_{x \in R_n} \quad (1)$$

By randomizing we get

$$F(X) = E[f(X)] \to \min_{x \in R_n} \quad (2)$$

where X is a random vector from R^n, {X} is a set of such random vectors, and $E[\cdot]$ is the expectation operator.

Problem 2 is equivalent to problem 1 in the sense that any realization of the random vector X^*, where X^* is a solution to 2, that has a nonzero probability, will be a solution to problem 1 (see [KAP] for proof).

Note that 2 is the stochastic optimization problem of the functional F(X) .

To study the gradient nature of the solution algorithms for problem 2, a variation of objective functional F(X) will be considered.

The suggested approach makes it possible to obtain optimization methods in systematic way, similar to the methodology adopted in smooth optimization. Derivation includes randomization of the original optimization problem, finding directional derivative for the randomized problem and choosing moving direction Y based on the condition that directional derivative in the direction of Y is being less or equal to 0.

Because of randomization, the expression for directional derivative doesn't contain the differential characteristics of the original function. We obtain the condition for selecting the direction of search Y in terms of its characteristics - conditional expectation. Conditional expectation is a vector function (or vector field) and can be decomposed (following the theorem of decomposition of the vector field) into the sum of the gradient of scalar function P and a function with zero divergence. P is called a potential function. As a result the original problem is reduced to optimization of the potential function, furthermore, the potential function is specific for each iteration step. Next, we arrive at partial differential equation that connects P and the original function. To define computational algorithms it is necessary to specify the dynamics of the random vectors. For example, the dynamics can be expressed in a form of densities. For certain class of distributions, for example normal distribution, the dynamics can be written in terms of expectation and covariance matrix. It is also possible to express the dynamics in mixed characteristics.

Expression for directional derivative

Derivative of objective functional F(X) in the direction of the random vector Y at the point X^0 (Gateaux derivative) is:

$$\delta_Y F(X^0) = \frac{d}{d\varepsilon} F(X^0 + \varepsilon Y)_{\varepsilon=0} = \frac{d}{d\varepsilon} F(X^\varepsilon) dx_{\varepsilon=0} = \frac{d}{d\varepsilon} \int f(X) p_{x^\varepsilon}(x)_{\varepsilon=0}$$

where density function of the random vector $X^\varepsilon = X^0 + \varepsilon Y$ may be expressed in terms of joint density function $p_{X^0,Y}(x,y)$ of X^0 and Y as follows:

$$p_{x^\varepsilon}(x) = \int_{R^n} p_{x^\varepsilon}(x - \varepsilon y, y) dy \quad (3)$$

The following relation (property of divergence) will be needed later

$$\frac{d}{d\varepsilon} p_{x^\varepsilon}(x - \varepsilon y, y) = (-\nabla_x p_{x^\varepsilon}(x,y), y) = -div_x(p_{x^\varepsilon}(x,y)y) \quad (4)$$

where (,) defines dot product.

Assuming differentiability of the integrals (for example, by selecting the appropriate $p_{x^\varepsilon}(x,y)$ and using 3, 4 we get

$$\delta_Y F(X^0) = [\frac{d}{d\varepsilon} \int_{R^n} \int_{R^n} f(x) p_{x^\varepsilon}(x - \varepsilon y, y) dx dy]_{\varepsilon=0} =$$

$$= [\frac{d}{d\varepsilon} \int_{R^n} f(x) \int_{R^n} p_{x^\varepsilon}(x - \varepsilon y, y) dx dy]_{\varepsilon=0} = [\int_{R^n} f(x) (\frac{d}{d\varepsilon} \int_{R^n} p_{x^\varepsilon}(x - \varepsilon y, y) dy) dx]_{\varepsilon=0} =$$

$$= \int_{R^n} f(x) (\int_{R^n} [\frac{d}{d\varepsilon} p_{x^\varepsilon}(x - \varepsilon y, y)]_{\varepsilon=0} dy) dx = -\int_{R^n} f(x) (\int_{R^n} [div_x(p_{x^\varepsilon}(x,y)y)] dy) dx =$$

$$-\int_{R^n} f(x) div_x [\int_{R^n} (p_{x^\varepsilon}(x,y)y) dy] dx$$

Using formula for conditional distribution $p_{Y/X^0=x}(y) = \frac{p_{x^\varepsilon y}(x,y)}{p_{x^\varepsilon}(x))}$,

where $p_{x^\varepsilon}(x) = \int_{R^n} p_{x^\varepsilon y}(x,u) du$

we get $\delta_Y F(X^0) = -\int_{R^n} f(x) div_x [p_{x^\varepsilon}(x) \int_{R^n} p_{Y/X^0=x}(y) y dy] dx$

Denote $\overline{y}(x) = \int_{R^n} y p_{Y/X^0=x}(y) dy = E[Y/X^0 = x]$

Taking into account normalization condition for density we arrive at the following expression for directional derivative:

$$\delta_Y F(X^0) = -\int_{R^n} (f(x) - C) div_x [p_{x^0}(x) \overline{y}(x)] dx$$

where C is arbitrary chosen constant

Considering solution to $\delta_Y F(X^0) \to \min_Y$ allows to obtain gradient-like algorithms for optimization that use only objective function values (do not use derivatives of objective function)

Potential function as a solution to Poisson's equation

Decomposing vector field $p_{x^0}(x)\overline{y}(x)$ into potential field $\nabla\varphi_0(x)$ and divergence-free component $W_0(x)$:

$$p_{x^0}(x)\overline{y}(x) = \nabla\phi_0(x) + W_0(x)$$

we arrive at Poisson's equation for potential function:

$$\Delta\varphi_0(x) = -L[f(x) - C] p_u(x)$$

where L is a constant

Solution to Poisson's equation approaching 0 at infinity may be written in the following form

$$\varphi_0(x) = \int_{R^n} E(x,\xi)[f(\xi) - C] p_u(\xi) d\xi$$

where $E(x,\xi)$ is a fundamental solution to Laplace's equation.

Then for potential component $\Delta\varphi_0(x)$ we have

$$\Delta\varphi_0(x) = -LE[\Delta_x E(x,u)(f(x) - C)]$$

To conclude, the representation for gradient-like direction is obtained. This direction maximizes directional derivative of the objective functional F(X). Therefore, this representation can be used for computing the gradient of the objective function f(x) using only its values. Gradient direction of the objective function f(x) is determined by the gradient of the potential function $\varphi_0(x)$, which, in turn, is determined by Poisson's equation.

Practical considerations

The dynamics of the expectation of objective function may be written in the space of random vectors as follows:

$$X_{N+1} = X_N + \alpha_{N+1} Y_{N+1}$$

where N - iteration number, Y^{N+1} - random vector that defines direction of move at (N+1)th iteration, α_{N+1} -step size on (N+1)th iteration. Y^{N+1} must be feasible at each iteration, i.e. the objective functional should decrease: $F(X^{N+1}) < (X^N)$. Applying expection to (12) and presenting $E[Y_{N+1}$ asconditional expectation $E_x E[Y/X]$ we get:

$$X_{N+1} = E[X_N] + \alpha_{N+1} E_{X^N} E[Y^{N+1}/X^N]$$

Replacing mathematical expectations $E[X_N]$ and $Y_{N+1}]$ with their estimates $\overline{E}^{N+1}$ and $\overline{y}(X^N)$ we get:

$$\overline{E}^{N+1} = \overline{E}^N + \alpha_{N+1} \overline{E}_{X^N}[\overline{y}(X^N)]$$

Note that expression for $\overline{y}(X^N)$ was obtained in the previos section up to certain parameters. By setting parameters to certain values we can obtain stochastic extensions of well known heuristics such as Nelder and Mead algorithm or Covariance Matrix Adaptation Evolution Strategy. In minpy library we use several common building blocks to create different algorithms. Customized algorithms may be defined by combining these common blocks and varying their parameters.

Main building blocks include computing center of mass of the sample points and finding newtonian potential.

Key takeaways, example algorithm, and code organization

Many industry professionals and researchers utilize mathematical optimization packages to search for better solutions of their problems. Examples of such problem include minimization of free energy in physical system [FW], robot gait optimization from robotics [PHS], designing materials for 3D printing [ZM], [TMAACBA], wine production [CTC], [CWC], optimizing chemical reactions [VNJT]. These problems may involve "black box optimization", where the structure of the objective function is unknown and is revealed through a small sequence of expensive trials. Software implementations for these methods become more user friendly. As a rule, however, certain modeling skills are needed to formulate real world problem in a way suitable for applying software package. Moreover, selecting optimization method appropriate for the model is a challenging task. Our educational software helps users of such optimization packages and may be considered as a companion to them. The focus of our software is on transparency of the methods rather than on efficiency. A principal benefit of our software is the unified approach for constructing algorithms whereby any other algorithm is obtained from the generalized algorithm by changing certain parameters. Well known heuristic algorithms such as Nelder and Mead (NM) algorithm may be obtained using this generalized approach, as well as new algorithms. Although some derivative-free optimization packages (matlab global optimization toolbox, Tensorflow Probability optimizers, Excel Evolutionary Solver, scikit-learn Stochastic Gradient Descent class, scipy.optimize.shgo method) put a lot of effort in transparency and educational value, they don't have the same level of flexibility and generality as our system. An example of educational-only optimization software is [SAS]. It is limited to teach Particle Swarm Optimization.

The code is organized in such a way that it allows to pair the algorithm with objective function. The new algorithm may be implmented as method of class Minimize. Newly created algorithm can be paired with test objectivve function supplied with a library or with externally supplied objective function (implemented in separate python module). New algorithms can be made more or less universal, that is, may have different number of parameters that user can specify. For example, it is possible to create Nelder and Mead algorithm (NM) using basic modules, and this would be an example of the most specific algorithm. It is also possible to create Stochastic Extention of NM (more generic than classic NM, similar to Simplicial Homology Global Optimisation [ESF] method) and with certain settings of adjustable parameters it may work identical to classic NM. Library repository may be found here: https://github.com/nadiakap/MinPy_edu

The following algorithms demonstrate steps similar to steps of Nelder and Mead algorithm (NM) but select only those points with objective function values smaller or equal to mean level of objective funtion. Such an improvement to NM assures its convergence [KPP]. Unlike NM, they are derived from the generic approach. First variant (NM-stochastic) resembles NM but corrects some of its drawbacks, and second variant (NM-nonlocal) has some similarity to random search as well as to NM and helps to resolve some other issues of classical NM algorithm.

Steps of NM-stochastic:

1) Initialize the search by generating $K \geq n$ separate realizations of u_0^i, i=1,..K of the random vector U_0, and set $m_0 = \frac{1}{K}\sum_{i=0}^{K} u_0^i$
2) On step j = 1, 2, ...

a.Compute the mean level $c_{j-1} = \frac{1}{K}\sum_{i=1}^{K} f(u_{j-1}^i)$

b.Calculate new set of vertices:

$$u_j^i = m_{j-1} + \varepsilon_{j-1}(f(u_{j-1}^i) - c_{j-1})\frac{m_{j-1} - u_{j-1}^i}{||m_{j-1} - u_{j-1}^i||^n}$$

c.Set $m_j = \frac{1}{K}\sum_{i=0}^{K} u_j^i$

d.Adjust the step size ε_{j-1} so that $f(m_j) < f(m_{j-1})$. If approximate ε_{j-1} cannot be obtained within the specified number of trails, then set $m_k = m_{j-1}$

e.Use sample standard deviation as termination criterion:

$$D_j = (\frac{1}{K-1}\sum_{i=1}^{K}(f(u_j^i) - c_j)^2)^{1/2}$$

Note that classic simplex search methods do not use values of objective function to calculate reflection/expantion/contraction coefficients. Those coefficients are the same for all vertices, whereas in NM-stochastic the distance each vertex will travel depends on the difference between objective function value and average value across all vertices $(f(u_j^i) - c_j)$. NM-stochastic shares the following drawbacks with classic simplex methods: a. simlex may collapse into a nearly degenerate figure, and usually proposed remedy is to restart the simlex every once in a while, b. only initial vertices are randomly generated, and the path of all subsequent vertices is deterministic. Next variant of the algorithm (NM-nonlocal) maintains the randomness of vertices on each step, while adjusting the distribution of U_0 to mimic the pattern of the modified vertices. The corrected algorithm has much higher exploration power than the first algorithm (similar to the exploration power of random search algorithms), and has exploitation power of direct - search algorithms.

Steps of NM - nonlocal

1) Choose a starting point x_0 and set $m_0 = x_0$.

2. On step j = 1, 2, ... Obtain K separate realizations of u_i^i, i=1,..K of the random vector U_j

a.Compute $f(u_{j-1}^i), j = 1, 2, ..K$, and the sample mean level

$$c_{j-1} = \frac{1}{K}\sum_{i=1}^{K} f(u_{j-1}^i)$$

b.Generate the new estimate of the mean:

$$m_j = m_{j-1} + \varepsilon_j \frac{1}{K}\sum_{i=1}^{K}[(f(u_j^i) - c_j)\frac{m_{j-1} - u_j^i}{||m_{j-1} - u_j^i||^n}]$$

Adjust the step size ε_{j-1} so that $f(m_j) < f(m_{j-1})$. If approximate ε_{j-1} cannot be obtained within the specified number of trails, then set $m_k = m_{j-1}$

c.Use sample standard deviation as termination criterion

$$D_j = (\frac{1}{K-1}\sum_{i=1}^{K}(f(u_j^i) - c_j)^2)^{1/2}$$

REFERENCES

[KAP] Kaplinskii, A.I.,Pesin, A.M.,Propoi, A.I.(1994). Analysis of search methods of optimization based on potential theory. I: Nonlocal properties. Automation and Remote Control. Volume 55, N.9, Part 2, September, pp.1316-1323 (rus. pp.97-105), 1994

[KP] Kaplinskii, A.I. and Propoi, A.I., Nonlocal Optimization Methods ofthe First Order Based on Potential Theory, Automation and Remote Control. Volume 55, N.7, Part 2, July, pp.1004-1011 (rus. pp.97-102), 1994

[KPP] Kaplinskii, A.I., Pesin, A.M.,Propoi, A.I. Analysis of search methods of optimization based on potential theory. III: Convergence of methods. Automation and remote Control, Volume 55, N.11, Part 1, November, pp.1604-1610 (rus. pp.66-72), 1994.

[NBAG] Nikhil Bansal, Anupam Gupta, Potential-function proofs for gradient methods, Theory of Computing, Volume 15, (2019) Article 4 pp. 1-32, https://doi.org/10.4086/toc.2019.v015a004

[ATFB] Adrien Taylor, Francis Bach, Stochastic first-order methods: non-asymptotic and computer-aided analyses via potential functions, arXiv:1902.00947 [math.OC], 2019, https://doi.org/10.48550/arXiv.1902.00947

[ZALO] Zeyuan Allen-Zhu and Lorenzo Orecchia, Linear Coupling: An Ultimate Unification of Gradient and Mirror Descent, Innovations in Theoretical Computer Science Conference (ITCS), 2017, pp. 3:1-3:22, https://doi.org/10.4230/LIPIcs.ITCS.2017.3

[BIS] Berthold Immanuel Schmitt, Convergence Analysis for Particle Swarm Optimization, FAU University Press, 2015

[FGSB] FJuan Frausto-Solis, Ernesto Liñán-García, Juan Paulo Sánchez-Hernández, J. Javier González-Barbosa, Carlos González-Flores, Guadalupe Castilla-Valdez, Multiphase Simulated Annealing Based on Boltzmann and Bose-Einstein Distribution Applied to Protein Folding Problem, Advances in Bioinformatics, Volume 2016, Article ID 7357123, https://doi.org/10.1155/2016/7357123

[GLUQ] Gong G., Liu, Y., Qian M, Simulated annealing with a potential function with discontinuous gradient on R^d, Ici. China Ser. A-Math. 44, 571-578, 2001, https://doi.org/10.1007/BF02876705

[PHS] Valdez, S.I., Hernandez, E., Keshtkar, S. (2020). A Hybrid EDA/Nelder-Mead for Concurrent Robot Optimization. In: Madureira, A., Abraham, A., Gandhi, N., Varela, M. (eds) Hybrid Intelligent Systems. HIS 2018. Advances in Intelligent Systems and Computing, vol 923. Springer, Cham. https://doi.org/10.1007/978-3-030-14347-3_20

[FW] Fan, Yi & Wang, Pengjun & Heidari, Ali Asghar & Chen, Huiling & HamzaTurabieh, & Mafarja, Majdi, 2022. "Random reselection particle swarm optimization for optimal design of solar photovoltaic modules," Energy, Elsevier, vol. 239(PA), https://doi.org/10.1016/j.energy.2021.121865

[VNJT] Fath, Verena, Kockmann, Norbert, Otto, Jürgen, Röder, Thorsten, Self-optimising processes and real-time-optimisation of organic syntheses in a microreactor system using Nelder–Mead and design of experiments, React. Chem. Eng., 2020,5, 1281-1299, https://doi.org/10.1039/D0RE00081G

[ZM] Plüss, T.; Zimmer, F.; Hehn, T.; Murk, A. Characterisation and Comparison of Material Parameters of 3D-Printable Absorbing Materials. Materials 2022, 15, 1503. https://doi.org/10.3390/ma15041503

[TMAACBA] Thoufeili Taufek, Yupiter H.P. Manurung, Mohd Shahriman Adenan, Syidatul Akma, Hui Leng Choo, Borhen Louhichi, Martin Bednardz, and Izhar Aziz.3D Printing and Additive Manufacturing, 2022, http://doi.org/10.1089/3dp.2021.0197

[CTC] Vismara, P., Coletta, R. & Trombettoni, G. Constrained global optimization for wine blending. Constraints 21, 597–615 (2016), https://doi.org/10.1007/s10601-015-9235-5

[CWC] Terry Hui-Ye Chiu, Chienwen Wu, Chun-Hao Chen, A Generalized Wine Quality Prediction Framework by Evolutionary Algorithms, International Journal of Interactive Multimedia and Artificial Intelligence, Vol. 6, Nº7,2021, https://doi.org/10.9781/ijimai.2021.04.006

[KHNT] Pascal Kerschke, Holger H. Hoos, Frank Neumann, Heike Trautmann; Automated Algorithm Selection: Survey and Perspectives. Evol Comput 2019; 27 (1): 3–45, https://doi.org/10.1162/evco_a_00242

[SAS] Leandro dos Santos Coelho, Cezar Augusto Sierakowski, A software tool for teaching of particle swarm optimization fundamentals, Advances in Engineering Software, Volume 39, Issue 11, 2008, Pages 877-887, ISSN 0965-9978, https://doi.org/10.1016/j.advengsoft.2008.01.005.

[ESF] Endres, S.C., Sandrock, C. & Focke, W.W. A simplicial homology algorithm for Lipschitz optimisation. J Glob Optim 72, 181–217 (2018), https://doi.org/10.1007/s10898-018-0645-y