The Journal of Open Source Software

Limbo: A Flexible High-performance Library for Gaussian Processes modeling and Data-Efficient Optimization

Antoine Cully[1], **Konstantinos Chatzilygeroudis**[2,3,4], **Federico Allocati**[2,3,4], **and Jean-Baptiste Mouret**[2,3,4]

1 Personal Robotics Lab, Imperial College London, London, United Kingdom **2** Inria Nancy Grand - Est, Villers-lès-Nancy, France **3** CNRS, Loria, UMR 7503, Vandœuvre-lès-Nancy, France **4** Université de Lorraine, Loria, UMR 7503, Vandœuvre-lès-Nancy, France

DOI: 10.21105/joss.00544

Software

- Review
- Repository
- Archive

Submitted: 22 January 2018
Published: 23 January 2018

Summary

Limbo (LIbrary for Model-Based Optimization) is an open-source C++11 library for Gaussian Processes and data-efficient optimization (e.g., Bayesian optimization, see (Shahriari et al. 2016)) that is designed to be both highly flexible and very fast. It can be used as a state-of-the-art optimization library or to experiment with novel algorithms with "plugin" components. Limbo is currently mostly used for data-efficient policy search in robot learning (Lizotte et al. 2007) and online adaptation because computation time matters when using the low-power embedded computers of robots. For example, Limbo was the key library to develop a new algorithm that allows a legged robot to learn a new gait after a mechanical damage in about 10-15 trials (2 minutes) (Cully et al. 2015), and a 4-DOF manipulator to learn neural networks policies for goal reaching in about 5 trials (Chatzilygeroudis et al. 2017).

The implementation of Limbo follows a policy-based design (Alexandrescu 2001) that leverages C++ templates: this allows it to be highly flexible without the cost induced by classic object-oriented designs (Driesen and Hölzle 1996) (cost of virtual functions). The regression benchmarks[1] show that the query time of Limbo's Gaussian processes is several orders of magnitude better than the one of GPy (a state-of-the-art Python library for Gaussian processes[2]) for a similar accuracy (the learning time highly depends on the optimization algorithm chosen to optimize the hyper-parameters). The black-box optimization benchmarks[3] demonstrate that Limbo is about 2 times faster than BayesOpt (a C++ library for data-efficient optimization, (Martinez-Cantin 2014)) for a similar accuracy and data-efficiency. In practice, changing one of the components of the algorithms in Limbo (e.g., changing the acquisition function) usually requires changing only a template definition in the source code. This design allows users to rapidly experiment and test new ideas while keeping the software as fast as specialized code.

Limbo takes advantage of multi-core architectures to parallelize the internal optimization processes (optimization of the acquisition function, optimization of the hyper-parameters of a Gaussian process) and it vectorizes many of the linear algebra operations (via the Eigen 3 library[4] and optional bindings to Intel's MKL). To keep the library lightweight, most of the optimizers in Limbo are wrappers around external optimization libraries:

[1] http://www.resibots.eu/limbo/reg_benchmarks.html
[2] https://sheffieldml.github.io/GPy/
[3] http://www.resibots.eu/limbo/bo_benchmarks.html
[4] http://eigen.tuxfamily.org/

- NLOpt[5] (which provides many local, global, gradient-based, gradient-free algorithms)
- libcmaes[6] (which provides the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES), and several variants of it (Hansen and Ostermeier 1996))
- a few other algorithms that are implemented in Limbo (in particular, RPROP (Riedmiller and Braun 1993), which is a gradient-based optimization algorithm)

The library is distributed under the CeCILL-C license[7] via a GitHub repository,[8] with an extensive documentation[9] that contains guides, examples, and tutorials. The code is standard-compliant but it is currently mostly developed for GNU/Linux and Mac OS X with both the GCC and Clang compilers. New contributors can rely on a full API reference, while their developments are checked via a continuous integration platform (automatic unit-testing routines).

Limbo is currently used in the ERC project ResiBots[10], which is focused on data-efficient trial-and-error learning for robot damage recovery, and in the H2020 projet PAL[11], which uses social robots to help coping with diabetes. It has been instrumental in many scientific publications since 2015 (Cully et al. 2015) (Chatzilygeroudis, Vassiliades, and Mouret 2018) (Tarapore et al. 2016) (Chatzilygeroudis et al. 2017) (Pautrat, Chatzilygeroudis, and Mouret 2018) (Chatzilygeroudis and Mouret 2018)

Acknowledgments

This work received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (GA no. 637972, project "ResiBots") and from the European Commission through the H2020 projects AnDy (GA no. 731540) and PAL (GA no. 643783).

References

Alexandrescu, Andrei. 2001. *Modern C++ Design: Generic Programming and Design Patterns Applied*. Addison-Wesley.

Chatzilygeroudis, Konstantinos, and Jean-Baptiste Mouret. 2018. "Using Parameterized Black-Box Priors to Scale up Model-Based Policy Search for Robotics." In *International Conference on Robotics and Automation (ICRA)*.

Chatzilygeroudis, Konstantinos, Roberto Rama, Rituraj Kaushik, Dorian Goepp, Vassilis Vassiliades, and Jean-Baptiste Mouret. 2017. "Black-Box Data-efficient Policy Search for Robotics." In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vancouver, Canada. https://hal.inria.fr/hal-01576683.

Chatzilygeroudis, Konstantinos, Vassilis Vassiliades, and Jean-Baptiste Mouret. 2018. "Reset-free Trial-and-Error Learning for Robot Damage Recovery." *Robotics and Autonomous Systems*.

Cully, Antoine, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. 2015. "Robots That Can Adapt Like Animals." *Nature* 521 (7553). Nature Publishing Group:503–7.

[5] http://ab-initio.mit.edu/nlopt
[6] https://github.com/beniz/libcmaes
[7] http://www.cecill.info/index.en.html
[8] http://github.com/resibots/limbo
[9] http://www.resibots.eu/limbo
[10] http://www.resibots.eu
[11] http://www.pal4u.eu/

Driesen, Karel, and Urs Hölzle. 1996. "The Direct Cost of Virtual Function Calls in C++." In *ACM Sigplan Notices*, 31:306–23. 10. ACM.

Hansen, Nikolaus, and Andreas Ostermeier. 1996. "Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation." In *Proceedings of IEEE International Conference on Evolutionary Computation*, 312–17. IEEE.

Lizotte, Daniel J, Tao Wang, Michael H Bowling, and Dale Schuurmans. 2007. "Automatic Gait Optimization with Gaussian Process Regression." In *Proceedings of Ijcai*, 7:944–49.

Martinez-Cantin, Ruben. 2014. "BayesOpt: A Bayesian Optimization Library for Nonlinear Optimization, Experimental Design and Bandits." *Journal of Machine Learning Research* 15:3915–9.

Pautrat, Rémi, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. 2018. "Bayesian Optimization with Automatic Prior Selection for Data-Efficient Direct Policy Search." In *International Conference on Robotics and Automation (ICRA)*.

Riedmiller, Martin, and Heinrich Braun. 1993. "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm." In *IEEE International Conference on Neural Networks*, 586–91. IEEE.

Shahriari, Bobak, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. 2016. "Taking the Human Out of the Loop: A Review of Bayesian Optimization." *Proceedings of the IEEE* 104 (1). IEEE:148–75.

Tarapore, Danesh, Jeff Clune, Antoine Cully, and Jean-Baptiste Mouret. 2016. "How Do Different Encodings Influence the Performance of the MAP-Elites Algorithm?" In *The 18th Annual conference on Genetic and evolutionary computation (GECCO'14)*. ACM. https://doi.org/10.1145/2908812.2908875.