The Journal of Open Source Software

pyGPGO: Bayesian Optimization for Python

José Jiménez[1] **and Josep Ginebra**[2]

1 Computational Biophysics Laboratory, Universitat Pompeu Fabra, Parc de Recerca Biomèdica de Barcelona, Carrer del Dr. Aiguader 88. Barcelona 08003, Spain. **2** Department of Statistics and Operations Research. Universitat Politècnica de Catalunya (UPC). Av. Diagonal 647, Barcelona 08028, Spain.

DOI: 10.21105/joss.00431

Software

- Review
- Repository
- Archive

Summary

Bayesian optimization has risen over the last few years as a very attractive method to optimize expensive to evaluate, black box, derivative-free and possibly noisy functions (Shahriari et al. 2016). This framework uses *surrogate models*, such as the likes of a Gaussian Process (Rasmussen and Williams 2004) which describe a prior belief over the possible objective functions in order to approximate them. The procedure itself is inherently sequential: our function is first evaluated a few times, a surrogate model is then fit with this information, which will later suggest the next point to be evaluated according to a predefined *acquisition function*. These strategies typically aim to balance exploitation and exploration, that is, areas where the posterior mean or variance of our surrogate model are high respectively.

These strategies have recently grabbed the attention of machine learning researchers over simpler black-box optimization strategies, such as grid search or random search (Bergstra James and Bengio Yoshua 2012). It is specially interesting in areas such as automatic machine-learning hyperparameter optimization (Snoek, Larochelle, and Adams 2012), A/B testing (Chapelle and Li 2011) or recommender systems (Vanchinathan et al. 2014), among others. Furthermore, the framework is entirely modular; there are many choices a user could take regarding the design of the optimization procedure: choice of surrogate model, covariance function, acquisition function behaviour or hyperparameter treatment, to name a few.

Here we present *pyGPGO* , an open-source Python package for Bayesian Optimization, which embraces this modularity in its design. While additional Python packages exist for the same purpose, either they are restricted for non-commercial applications (Snoek 2012), implement a small subset of the features (Yelp 2014), or do not provide a modular interface (team. 2016). *pyGPGO* on the other hand aims to provide the highest degree of freedom in the design and inference of a Bayesian optimization pipeline, while being feature-wise competitive with other existing software. *pyGPGO* currently supports:

- Different surrogate models: Gaussian Processes, Student-t Processes, Random Forests (& variants) and Gradient Boosting Machines.
- Most usual covariance function structures, as well as their derivatives: squared exponential, Matèrn, gamma-exponential, rational-quadratic, exponential-sine and dot-product kernel.
- Several acquisition function behaviours: probability of improvement, expected improvement, upper confidence bound and entropy-based, as well as their integrated versions.
- Type II maximum-likelihood estimation of covariance hyperparameters.
- MCMC sampling for the full-bayesian treatment of hyperparameters (via `pyMC3` (Salvatier, Wiecki, and C. 2016))

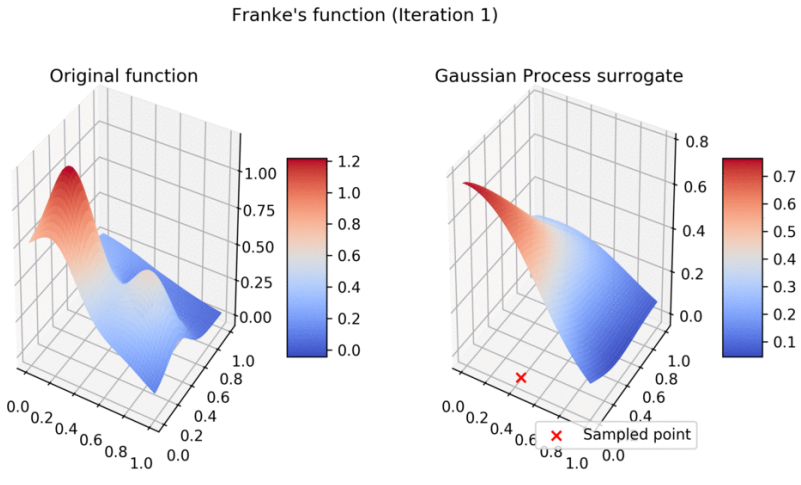

Figure 1: pyGPGO in action.

pyGPGO is MIT-licensed and can be retrieved from both GitHub and PyPI, with extensive documentation available at ReadTheDocs. *pyGPGO* is built on top of other well known packages of the Python scientific ecosystem as dependencies, such as numpy, scikit-learn, pyMC3 and theano.

Future work

pyGPGO is an ongoing project, and as such there are several improvements that will be tackled in the near future:

- Support for linear combinations of covariance functions, with automatic gradient computation.
- Support for more diverse acquisition functions, such as Predictive Entropy Search (Hernández-Lobato, Hoffman, and Ghahramani 2014).
- A class for constrained Bayesian Optimization is planned for the near future. (Gardner et al. 2014)

References

Bergstra James, James, and Umontrealca Bengio Yoshua. 2012. "Random Search for Hyper-Parameter Optimization." *Journal of Machine Learning Research* 13: 281–305. doi:10.1162/153244303322533223.

Chapelle, Olivier, and Lihong Li. 2011. "An Empirical Evaluation of Thompson Sampling." *Advances in Neural Information Processing Systems*, 2249——2257. http://explo.cs.ucl.ac.uk/wp-content/uploads/2011/05/An-Empirical-Evaluation-of-Thompson-Sampling-Chap pdf.

Gardner, Jacob R., Matt J. Kusner, Zhixiang Eddie Xu, Kilian Q. Weinberger, and John P. Cunningham. 2014. "Bayesian Optimization with Inequality Constraints." *Proceedings of the 31st International Conference on Machine Learning* 32: 937–45.

Hernández-Lobato, José Miguel, Matthew W Hoffman, and Zoubin Ghahramani. 2014. "Predictive Entropy Search for Efficient Global Optimization of Black-box Functions."

Advances in Neural Information Processing Systems 28, 1–9. https://jmhldotorg.files.wordpress.com/2014/10/pes-final.pdf.

Rasmussen, Carl E., and Christopher K. I. Williams. 2004. *Gaussian processes for machine learning.* Vol. 14. 2. doi:10.1142/S0129065704001899.

Salvatier, J, TV Wiecki, and Fonnesbeck C. 2016. "Probabilistic programming in Python using PyMC3." *PeerJ Computer Science*. https://doi.org/10.7717/peerj-cs.55.

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): 148–75. doi:10.1109/JPROC.2015.2494218.

Snoek, Jasper. 2012. "Spearmint." https://github.com/HIPS/Spearmint.

Snoek, Jasper, Hugo Larochelle, and Rp Adams. 2012. "Practical Bayesian optimization of machine learning algorithms." *Advances in Neural Information …*, 1–9. doi:2012arXiv1206.2944S.

team., The scikit-optimize. 2016. "Scikit-Optimize." https://github.com/scikit-optimize/scikit-optimize.

Vanchinathan, Hastagiri P., Isidor Nikolic, Fabio De Bona, and Andreas Krause. 2014. "Explore-exploit in top-N recommender systems via Gaussian processes." In *Proceedings of the 8th Acm Conference on Recommender Systems - Recsys '14*, 225–32. doi:10.1145/2645710.2645733.

Yelp. 2014. "MOE." https://github.com/Yelp/MOE.