The Journal of Open Source Software

mfpml: Multi-fidelity probabilistic machine learning toolkit

Jiaxiang Yi [1¶] and **Ji Cheng** [2]

1 Faculty of Mechanical Engineering, Delft University of Technology, the Netherlands **2** City University of Hong Kong, Hong Kong ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Vangelis Kourlitis

Reviewers:

- @salrm8
- @jungtaekkim

Submitted: 24 December 2024
Published: unpublished

Summary

The `mfpml` (multi-fidelity probabilistic machine learning) package provides a Python platform for implementing classic single- and multi-fidelity Bayesian machine learning surrogates and applying them to Bayesian optimization. Although numerous methods have been developed in the domain of multi-fidelity machine learning (Giselle Fernández-Godino, 2023), no open-source software offers a comprehensive suite of tools for this purpose. This package addresses this gap by providing a platform to replicate existing single- and multi-fidelity Bayesian methods based on Gaussian process regression. Furthermore, it serves as a handy tool for developing new methods in the field of multi-fidelity probabilistic machine learning.

Figure 1: Logo of mfpml (`mfpml`).

Statement of Need

`mfpml` is written in Python and depends on a few third-party packages, such as NumPy (Harris et al., 2020) and SciPy (Virtanen et al., 2020). It includes detailed notebooks and autogenerated Sphinx documentation, allowing users to replicate existing methods and develop new ones with ease. Specifically, it provides essential modules for building machine learning models, including design of experiments, benchmark problems, models, and optimization.

Key features of `mfpml` include:

1. **Basic Methods**: Fundamental implementations of popular methods, such as Gaussian process regression (Rasmussen & Williams, 2005), Co-Kriging (Forrester et al., 2007), and corresponding extensions (Han & Görtz, 2012).
2. **Advanced Methods**: Advanced techniques for Bayesian optimization (Jones et al., 1998), including single-fidelity and multi-fidelity optimization.
3. **Future Development**: Ongoing work includes adding constrained optimization and multi-objective optimization methods, which will be included in future versions.

Acknowledgements

We thank the Python code of conduct provided by BessaGroup and acknowledge the support from our institutions.

References

Forrester, A. I. J., Sóbester, A., & Keane, A. J. (2007). Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *463*(2088), 3251–3269. https://doi.org/10.1098/rspa.2007.1900

Giselle Fernández-Godino, M. (2023). Review of multi-fidelity models. *Advances in Computational Science and Engineering*, *1*(4), 351–400. https://doi.org/10.3934/acse.2023015

Han, Z.-H., & Görtz, S. (2012). Hierarchical kriging model for variable-fidelity surrogate modeling. *AIAA Journal*, *50*(9), 1885–1896. https://doi.org/10.2514/1.J051354

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, *13*(4), 455–492. https://doi.org/10.1023/A:1008306431147

Rasmussen, C. E., & Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. The MIT Press. https://doi.org/10.7551/mitpress/3206.001.0001

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2