The Journal of Open Source Software

GPJax: A Gaussian Process Framework in JAX

Thomas Pinder[1¶] **and Daniel Dodd**[2]

1 Department of Mathematics and Statistics, Lancaster University, United Kingdom **2** STOR-i Centre for Doctoral Training, Lancaster University, United Kingdom ¶ Corresponding author

DOI: 10.21105/joss.04455

Software

- Review
- Repository
- Archive

Editor: Dan Foreman-Mackey

Reviewers:

- @gpleiss
- @theorashid

Submitted: 24 May 2022
Published: 23 July 2022

Summary

Gaussian processes (GPs, Rasmussen & Williams, 2006) are Bayesian nonparametric models that have been successfully used in applications such as geostatistics (Matheron, 1963), Bayesian optimisation (Mockus et al., 1978), and reinforcement learning (Deisenroth & Rasmussen, 2011). `GPJax` is a didactic GP library targeted at researchers who wish to develop novel GP methodology. The scope of `GPJax` is to provide users with a set of composable objects for constructing GP models that closely resemble the underlying maths that one would write on paper. Furthermore, by the virtue of being written in JAX (Bradbury et al., 2018), `GPJax` natively supports CPUs, GPUs and TPUs through efficient compilation to XLA, automatic differentiation and vectorised operations. Consequently, `GPJax` provides a modern GP package that can effortlessly be tailored, extended and interleaved with other libraries to meet the individual needs of researchers and scientists.

Statement of Need

From both an applied and methodological perspective, GPs are widely employed in the statistics and machine learning communities. High-quality software packages that promote GP modelling are accountable for much of their success. However, there currently exists a gap within the JAX ecosystem for a Gaussian process package to be developed that incorporates scalable inference techniques. `GPJax` seeks to resolve this.

`GPJax` has been carefully tailored to amalgamate with the JAX ecosystem. For efficient Markov Chain Monte Carlo inference, `GPJax` can utilise samplers from BlackJax (BlackJax, 2021) and TensorFlow Probability (Abadi et al., 2016). For gradient-based optimisation, `GPJax` integrates seamlessly with Optax (Babuschkin et al., 2020), providing a vast suite of optimisers and learning rate schedules. To efficiently represent probability distributions, `GPJax` leverages Distrax (Babuschkin et al., 2020) and TensorFlow Probability (Abadi et al., 2016). To combine GPs with deep learning methods, `GPJax` can incorporate the functionality provided within Haiku (Babuschkin et al., 2020). The `GPJax` documentation includes examples of each of these integrations.

The foundation of each abstraction given in `GPJax` is a Chex (Babuschkin et al., 2020) dataclass object. These require significantly less boilerplate code than regular Python classes, leading to a more readable codebase. Moreover, Chex dataclasses are registered as PyTree nodes, facilitating the applications of JAX operations such as just-in-time compilation and automatic differentiation to any `GPJax` object.

The intimacy between `GPJax` and the underlying maths also makes `GPJax` an excellent package for people new to GP modelling. Having the ability to easily cross-reference the contents of a textbook with the code that one is writing is invaluable when trying to build an intuition for a new statistical method. We further support this effort in `GPJax` through documentation that provides detailed explanations of the operations conducted within each notebook.

Wider Software Ecosystem

Within the Python community, the three most popular packages for GP modelling are GPFlow (Matthews et al., 2017), GPyTorch (Gardner et al., 2018), and GPy (GPy, 2012). Despite these packages being indispensable tools for the community, none support integration with a JAX-based workflow. On the other hand, BayesNewton (Wilkinson et al., 2021) and TinyGP (Foreman-Mackey, 2021) packages utilise a Jax backend. However, BayesNewton is designed on top of ObJax (Objax Developers, 2020), making integration with the broader Jax ecosystem challenging. Meanwhile, TinyGP offers excellent integration with inference frameworks such as NumPyro (Phan et al., 2019) but does not yet support inducing points frameworks (e.g., Hensman et al., 2013). `GPJax` exists to resolve these issues. Furthermore, modern research from the GP literature, graph kernels (Borovitskiy et al., 2021) and Wasserstein barycentres for GPs (Mallasto & Feragen, 2017), for example, are supported within `GPJax` but absent from these packages. Finally, the Stheno package (Bruinsma, 2022) supports a JAX backend along with TensorFlow, PyTorch and Numpy. Whilst this integrates GPs into an extensive JAX workflow, `GPJax` has the advantage of being a pure JAX codebase, whereas Stheno requires using a custom linear algebra framework.

For completeness, packages written for languages other than Python include GPML (Rasmussen & Nickisch, 2010) and GPStuff (Vanhatalo et al., 2013) in MATLAB. An R port also exists for GPStuff. Within Julia, there exists GaussianProcesses.jl (Fairbrother et al., 2022), AugmentedGaussianProcesses.jl (Galy-Fajou et al., 2020) and Stheno.jl (Tebbutt W, 2022).

GP implementations are available in numerous modern probabilistic programming languages such as NumPyro (Phan et al., 2019), Stan (Carpenter et al., 2017), and PyMC (Salvatier et al., 2016).

External Usage

Two recent research papers (Pinder et al., 2021, 2022) utilise the graph kernel functionality provided by `GPJax`.

Acknowledgments

As an open-source project, `GPJax` has benefitted from contributions made by the wider community. We especially thank Juan Emmanuel Johnson and are grateful for the thoughts and advice from the wider GP community.

Funding Statement

TP is supported by the Data Science for the Natural Environment project (EPSRC grant number EP/R01860X/1). DD is supported by the STOR-i Centre for Doctoral Training (EPSRC grant number EP/S022252/1) and the Research Hub for Transforming Energy Infrastructure through Digital Engineering (ARC grant number IH200100009).

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., & others. (2016). TensorFlow: A system for large-scale machine learning. *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 265–283.

Babuschkin, I., Baumli, K., Bell, A., Bhupatiraju, S., Bruce, J., Buchlovsky, P., Budden, D., Cai, T., Clark, A., Danihelka, I., Fantacci, C., Godwin, J., Jones, C., Hennigan, T., Hessel, M., Kapturowski, S., Keck, T., Kemaev, I., King, M., … Viola, F. (2020). *The DeepMind JAX Ecosystem*. http://github.com/deepmind

BlackJax. (2021). BlackJAX; a sampling library designed for ease of use, speed and modularity. In *GitHub repository*. GitHub. https://github.com/blackjax-devs/blackjax/

Borovitskiy, V., Azangulov, I., Terenin, A., Mostowsky, P., Deisenroth, M., & Durrande, N. (2021). Matérn Gaussian processes on graphs. *International Conference on Artificial Intelligence and Statistics*, 2593–2601.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). *JAX: Composable transformations of Python+NumPy programs* (Version 0.2.5) [Computer software]. http://github.com/google/jax

Bruinsma, W. (2022). Stheno. In *GitHub repository*. GitHub. https://github.com/wesselb/stheno

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., & Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software*, *76*(1).

Deisenroth, M., & Rasmussen, C. E. (2011). PILCO: A model-based and data-efficient approach to policy search. *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 465–472.

Fairbrother, J., Nemeth, C., Rischard, M., Brea, J., & Pinder, T. (2022). GaussianProcesses.jl: A nonparametric Bayes package for the Julia language. *Journal of Statistical Software*, *102*, 1–36.

Foreman-Mackey, D. (2021). TinyGP. In *GitHub repository*. GitHub. https://github.com/dfm/tinygp

Galy-Fajou, T., Wenzel, F., & Opper, M. (2020). Automated augmented conjugate inference for non-conjugate Gaussian process models. In S. Chiappa & R. Calandra (Eds.), *Proceedings of the twenty third international conference on artificial intelligence and statistics* (Vol. 108, pp. 3025–3035). PMLR.http://proceedings.mlr.press/v108/galy-fajou20a.html

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., & Wilson, A. G. (2018). GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration. *Advances in Neural Information Processing Systems*, *31*.

GPy. (2012). GPy: A Gaussian process framework in Python. In *GitHub repository*. GitHub. http://github.com/SheffieldML/GPy

Hensman, J., Fusi, N., & Lawrence, N. D. (2013). Gaussian processes for big data. *arXiv Preprint arXiv:1309.6835*.

Mallasto, A., & Feragen, A. (2017). Learning from uncertain curves: The 2-Wasserstein metric for Gaussian processes. *Advances in Neural Information Processing Systems*, *30*.

Matheron, G. (1963). Principles of geostatistics. *Economic Geology*, *58*(8), 1246–1266.

Matthews, A. G. de G., Van Der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., & Hensman, J. (2017). GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, *18*(40), 1–6.

Mockus, J., Tiesis, V., & Zilinskas, A. (1978). The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, *2*(117-129), 2.

Objax Developers. (2020). *Objax* (Version 1.2.0). https://github.com/google/objax

Phan, D., Pradhan, N., & Jankowiak, M. (2019). Composable effects for flexible and accelerated probabilistic programming in NumPyro. *arXiv Preprint arXiv:1912.11554*.

Pinder, T., Turnbull, K., Nemeth, C., & Leslie, D. (2021). Gaussian processes on hypergraphs. *arXiv Preprint arXiv:2106.01982*.

Pinder, T., Turnbull, K., Nemeth, C., & Leslie, D. (2022). Street-level air pollution modelling with graph Gaussian processes. *ICLR: AI for Earth and Space Science*.

Rasmussen, C. E., & Nickisch, H. (2010). Gaussian processes for machine learning (GPML) toolbox. *The Journal of Machine Learning Research*, *11*, 3011–3015.

Rasmussen, C. E., & Williams, C. K. (2006). *Gaussian processes for machine learning*. MIT press Cambridge, MA.

Salvatier, J., Wiecki, T. V., & Fonnesbeck, C. (2016). Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, *2*, e55. https://doi.org/10.7717/peerj-cs.55

Tebbutt W, T. R., Bruinsma W. (2022). Stheno.jl. In *GitHub repository*. GitHub. https://github.com/JuliaGaussianProcesses/Stheno.jl

Vanhatalo, J., Riihimäki, J., Hartikainen, J., Jylänki, P., Tolvanen, V., & Vehtari, A. (2013). GPstuff: Bayesian modeling with Gaussian processes. *Journal of Machine Learning Research*.

Wilkinson, W. J., Särkkä, S., & Solin, A. (2021). Bayes-Newton methods for approximate Bayesian inference with PSD guarantees. *arXiv Preprint arXiv:2111.01721*.