The Journal of Open Source Software

UltraNest - a robust, general purpose Bayesian inference engine

Johannes Buchner[1, 2, 3, 4]

1 Max Planck Institute for Extraterrestrial Physics, Giessenbachstrasse, 85741 Garching, Germany. **2** Millenium Institute of Astrophysics, Vicuña MacKenna 4860, 7820436 Macul, Santiago, Chile **3** Pontificia Universidad Católica de Chile, Instituto de Astrofísica, Casilla 306, Santiago 22, Chile. **4** Excellence Cluster Universe, Boltzmannstr. 2, D-85748, Garching, Germany

DOI: 10.21105/joss.03001

Software

- Review
- Repository
- Archive

Editor: Frederick Boehm

Reviewers:

- @mattpitkin
- @ziatdinovmax

Submitted: 22 January 2021
Published: 02 April 2021

Summary

UltraNest is a general-purpose Bayesian inference package for parameter estimation and model comparison. It allows fitting arbitrary models specified as likelihood functions written in Python, C, C++, Fortran, Julia or R. With a focus on correctness and speed (in that order), UltraNest is especially useful for multi-modal or non-Gaussian parameter spaces, computational expensive models, in robust pipelines. Parallelisation to computing clusters and resuming incomplete runs is available.

Statement of need

When scientific models are compared to data, two tasks are important: 1) constraining the model parameters and 2) comparing the model to other models. While several open source Bayesian model fitting packages are available that can be easily tied to existing models, they are difficult to run such that the result is reliable and user interaction is minimized. A chicken-and-egg problem is that one does not know a priori the posterior distribution of a given likelihood, prior and data set, and cannot choose a sampler that performs well. For example, Markov Chain Monte Carlo convergence checks may suggest good results, while in fact another distant but important posterior peak has remained unseen. Current and upcoming large astronomical surveys require characterising a large number of highly diverse objects, which requires reliable analysis pipelines. This is what UltraNest was developed for.

Nested sampling (NS, Skilling, 2004) allows Bayesian inference on arbitrary user-defined likelihoods. Additional to computing parameter posterior samples, it also estimates the marginal likelihood ("evidence," Z). Bayes factors between two competing models $B = Z_1/Z_2$ are a measure of the relative prediction parsimony of the models, and form the basis of Bayesian model comparison. By performing a global scan of the parameter space from the worst to best fits to the data, NS also performs well in multi-modal settings.

In the last decade, several variants of NS have been developed. The variants relate to (1) how better and better fits are found while respecting the priors, (2) whether it is allowed to go back to worse fits and explore the parameter space more, and (3) diagnostics through tests and visualisations. UltraNest develops novel, state-of-the-art techniques for all of the above. They are especially remarkable for being free of tuning parameters and theoretically justified.

Currently available efficient NS implementations such as `MultiNest` (Feroz et al., 2009) and its open-source implementations rely on a heuristic algorithm which has shown biases when the likelihood contours are not ellipsoidal (Buchner, 2014; Nelson et al., 2020). UltraNest

The Journal of Open Source Software

instead implements better motivated self-diagnosing algorithms, and improved, conservative uncertainty propagation. In other words, UltraNest prioritizes robustness and correctness, and maximizes speed second. For potentially complex posteriors where the user is willing to invest computation for obtaining a gold-standard exploration of the entire posterior distribution in one run, UltraNest was developed.

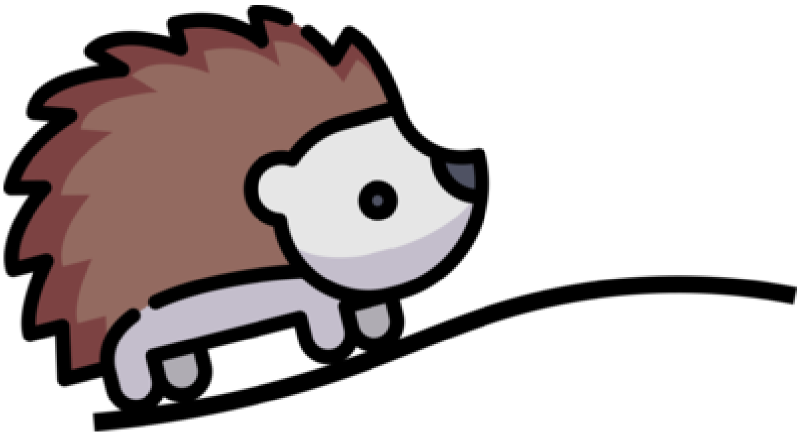

Figure 1: Logo of UltraNest; made by https://www.flaticon.com/authors/freepik

Method

NS methods are systematically reviewed in Buchner et al., submitted. The approaches used in UltraNest are highlighted there as well.

The basic outline of vanilla NS (see Skilling, 2004 for details) is as follows:

A set of N points are randomly drawn from the unit hypercube (u-space). A inverse cumulative prior transform converts these points to physical parameter units (v-space). The likelihood L evaluated for each point. In this population of live points, NS repeatedly replaces the current worst likelihood point through likelihood-constrained prior sampling (LRPS). At each iteration (represented by the removed, dead point), the prior space investigated shrinks by approximately $V_{i+1}/V_i = (N-1)/N$, starting from the entire prior volume $V_i = 1$. The dead point becomes a posterior sample with weight $w_i = L_i \times V_i$, yielding the posterior distribution and the evidence estimate $Z_i = \sum_{j=1}^{i} w_i$. The iteration procedure can terminate when the live points become unimportant, i.e. when $w_{live} = V_{i+1} \max_{i=1}^{N} L_{live,i} \ll Z_i$.

Reactive NS

Instead of iterating with a fixed array of live points, UltraNest uses a tree. The root of the tree represents the entire prior volume, and its child nodes are samples from the entire prior. A breadth-first search is run, which keeps a stack of the opened nodes sorted by likelihood. When encountering a node, attaching a child to it is decided by several criteria.

Reactive NS is a flexible generalisation of the Dynamic NS (Higson et al., 2019), which used a simple heuristic for identifying where to add more live points. The tree formulation of Reactive NS makes implementing error propagation and variable number of live points straight-forward.

Integration procedure

UltraNest computes conservative uncertainties on Z and the posterior weights. Several Reactive NS explorers are run which see only parts of the tree, specifically a bootstrapped subsample of the root edges. For each sample, each explorer estimates a weight (0 if it is blind to it), and an estimate of the evidence. The ensemble gives an uncertainty distribution.

The bootstrapped integrators are an evolution over single-bulk evidence uncertainty measures and includes the scatter in likelihoods (by bootstrapping) and volume estimates (by beta sampling; Skilling, 2004).

LRPS procedures in UltraNest

The live points all fulfill the current likelihood threshold, therefore they can be used to trace out the neighbourhood where a new, independent prior sample can be generated that also fulfills the threshold. Region-based sampling uses rejection sampling using constructed geometries.

UltraNest combines three region constructions, and uses their intersection: 1) MLFriends (Buchner, 2019), based on RadFriends (Buchner, 2014), 2) a bootstrapped single ellipsoid in u-space and 3) another bootstrapped single ellipsoid in v-space. The last one drastically helps when one parameter constraint scales with another, (e.g., funnels). UltraNest dynamically chooses whether to draw samples from the entire prior, the single u-space ellipsoid or MLFriends ellipsoids (accounting for overlaps), and filtered by the other constraints (including the transformed v-space ellipsoid).

Useful for high dimensional problems ($d > 20$), UltraNest supports several types of Monte Carlo random walks, including:

- Slice sampling (as in `Polychord`, Handley et al., 2015)
- Hit-and-run sampling
- Constrained Hamiltonian Monte Carlo with No-U turn sampling (similar to `NoGUTS`, Griffiths & Wales, 2019)

Features

- Run-time visualisation
- Posterior visualisations
- Diagnostic test of run quality
- MPI parallelisation
- Resuming
- Models written in Python, C, C++, Fortran, Julia (Schulz & Buchner, 2020), R, and Javascript (Buchner, 2018).

Extensive documentation is available.

Acknowledgements

I am very thankful to Fred Beaujean, Josh Speagle and J. Michael Burgess for insightful conversations.

References

Buchner, J. (2014). A statistical test for Nested Sampling algorithms. *Statistics and Computing*, 1–10. https://doi.org/10.1007/s11222-014-9512-y

Buchner, J. (2019). Collaborative Nested Sampling: Big Data versus Complex Physical Models. *131*(1004), 108005. https://doi.org/10.1088/1538-3873/aae7fc

Buchner, J. (2018). Ultranest-js. In *GitHub repository*. GitHub. https://github.com/JohannesBuchner/ultranest-js

Feroz, F., Hobson, M. P., & Bridges, M. (2009). MULTINEST: an efficient and robust Bayesian inference tool for cosmology and particle physics. *398*, 1601–1614. https://doi.org/10.1111/j.1365-2966.2009.14548.x

Griffiths, M., & Wales, D. J. (2019). Nested basin-sampling. *Journal of Chemical Theory and Computation*, *15*(12), 6865–6881. https://doi.org/10.1021/acs.jctc.9b00567

Handley, W. J., Hobson, M. P., & Lasenby, A. N. (2015). POLYCHORD: next-generation nested sampling. *453*, 4384–4398. https://doi.org/10.1093/mnras/stv1911

Higson, E., Handley, W., Hobson, M., & Lasenby, A. (2019). Dynamic nested sampling: an improved algorithm for parameter estimation and evidence calculation. *Statistics and Computing*, *29*(5), 891–913. https://doi.org/10.1007/s11222-018-9844-0

Nelson, B. E., Ford, E. B., Buchner, J., Cloutier, R., Díaz, R. F., Faria, J. P., Hara, N. C., Rajpaul, V. M., & Rukdee, S. (2020). Quantifying the Bayesian Evidence for a Planet in Radial Velocity Data. *159*(2), 73. https://doi.org/10.3847/1538-3881/ab5190

Schulz, O., & Buchner, J. (2020). UltraNest.jl. In *GitHub repository*. GitHub. https://github.com/bat/UltraNest.jl

Skilling, J. (2004). Nested sampling. *AIP Conference Proceedings*, *735*(1), 395. https://doi.org/10.1063/1.1835238