The Journal of Open Source Software

SBIAX: Density-estimation simulation-based inference in JAX.

Jed Homer[1*]

1 Ludwig-Maximilians-Universität München, Faculty for Physics, University Observatory, Scheinerstrasse 1, München, Deustchland. * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

In partnership with

AMERICAN ASTRONOMICAL SOCIETY

This article and software are linked with research article DOI 10.3847/xxxxx <- update this with the DOI from AAS once you know it., published in the Astrophysical Journal <- The name of the AAS journal..

Summary

In a typical Bayesian inference problem, the data likelihood is not known. However, in recent years, machine learning methods for density estimation can allow for inference using an estimator of the data likelihood. This likelihood is created with neural networks that are trained on simulations - one of the many tools for simulation based inference (SBI, Cranmer et al. (2020)). In such analyses, density-estimation simulation-based inference methods can derive a posterior, which typically involves

- simulating a set of data and model parameters $\{(\boldsymbol{\xi}, \boldsymbol{\pi})_0, ..., (\boldsymbol{\xi}, \boldsymbol{\pi})_N\}$,
- obtaining a measurement $\hat{\boldsymbol{\xi}}$,
- compressing the simulations and the measurements - usually with a neural network or linear compression - to a set of summaries $\{(\boldsymbol{x}, \boldsymbol{\pi})_0, ..., (\boldsymbol{x}, \boldsymbol{\pi})_N\}$ and $\hat{\boldsymbol{x}}$,
- fitting an ensemble of normalising flow or similar density estimation algorithms (e.g. a Gaussian mixture model),
- the optional optimisation of the parameters for the architecture and fitting hyperparameters of the algorithms,
- sampling the ensemble posterior (using an MCMC sampler if the likelihood is fit directly) conditioned on the datavector to obtain parameter constraints on the parameters of a physical model, $\boldsymbol{\pi}$.

`sbiax` is a code for implementing each of these steps. The code allows for Neural Likelihood Estimation (Alsing et al., 2019; Papamakarios, 2019) and Neural Posterior Estimation (Greenberg et al., 2019).

As shown in Homer et al. (2024), SBI is shown to successfully obtain the correct posterior widths and coverages given enough simulations which agree with the analytic solution.

Statement of need

Simulation-based inference (SBI) covers a broad class of statistical techniques such as Approximate Bayesian Computation (ABC), Neural Ratio Estimation (NRE), Neural Likelihood Estimation (NLE) and Neural Posterior Estimation (NPE). These techniques can derive posterior distributions conditioned of noisy data vectors in a rigorous and efficient manner. In particular, density-estimation methods have emerged as a promising method, given their efficiency, using generative models to fit likelihoods or posteriors directly using simulations.

In the field of cosmology, SBI is of particular interest due to complexity and non-linearity of models for the expectations of non-standard summary statistics of the large-scale structure, as well as the non-Gaussian noise distributions for these statistics. The assumptions required for the complex analytic modelling of these statistics as well as the increasing dimensionality of data returned by spectroscopic and photometric galaxy surveys limits the amount of information

JOSS
The Journal of Open Source Software

that can be obtained on fundamental physical parameters. Therefore, the study and research into current and future statistical methods for Bayesian inference is of paramount importance for the field of cosmology.

The software we present, `sbiax`, is designed to be used by machine learning and physics researchers for running Bayesian inferences using density-estimation SBI techniques. These models can be fit easily with multi-accelerator training and inference within the code. This code - written in `jax` (Bradbury et al., 2018) - allows for seemless integration of cutting edge generative models to SBI, including continuous normalising flows (Grathwohl et al., 2018), matched flows (Lipman et al., 2023), masked autoregressive flows (Papamakarios et al., 2018; Ward, 2024) and Gaussian mixture models - all of which are implemented in the code. The code features integration with the `optuna` (Akiba et al., 2019) hyperparameter optimisation framework which would be used to ensure consistent analyses, `blackjax` (Cabezas et al., 2024) for fast MCMC sampling and `equinox` (Kidger & Garcia, 2021) for neural network compression methods. The design of `sbiax` allows for new density estimation algorithms to be trained and sampled from.

Density estimation with normalising flows

The use of density-estimation in SBI has been accelerated by the advent of normalising flows. These models parameterise a change-of-variables $\boldsymbol{y} = f_\phi(\boldsymbol{x};\boldsymbol{\pi})$ between a simple base distribution (e.g. a multivariate unit Gaussian $\mathcal{G}[\boldsymbol{z}|\mathbf{0},\mathbf{I}]$) and an unknown distribution $q(\boldsymbol{x}|\boldsymbol{\pi})$ (from which we have simulated samples $\boldsymbol{x}$). Naturally, this is of particular importance in inference problems in which the likelihood is not known. The change-of-variables is fit from data by training neural networks to model the transformation in order to maximise the log-likelihood of the simulated data $\boldsymbol{x}$ conditioned on the parameters $\boldsymbol{\pi}$ of a simulator model. The mapping is expressed as

$$\boldsymbol{y} = f_\phi(\boldsymbol{x};\boldsymbol{\pi}),$$

where ϕ are the parameters of the neural network. The log-likelihood of the flow is expressed as

$$\log p_\phi(\boldsymbol{x}|\boldsymbol{\pi}) = \log \mathcal{G}[f_\phi(\boldsymbol{x};\boldsymbol{\pi})|\mathbf{0},\mathbb{I}] + \log\left|\mathbf{J}_{f_\phi}(\boldsymbol{x};\boldsymbol{\pi})\right|,$$

This density estimate is fit to a set of N simulation-parameter pairs $\{(\boldsymbol{\xi},\boldsymbol{\pi})_0, ..., (\boldsymbol{\xi},\boldsymbol{\pi})_N\}$ by minimising a Monte-Carlo estimate of the KL-divergence

$$\begin{aligned}
\langle D_{KL}(q||p_\phi)\rangle_{\boldsymbol{\pi}\sim p(\boldsymbol{\pi})} &= \int \mathrm{d}\boldsymbol{\pi}\; p(\boldsymbol{\pi}) \int \mathrm{d}\boldsymbol{x}\; q(\boldsymbol{x}|\boldsymbol{\pi}) \log \frac{q(\boldsymbol{x}|\boldsymbol{\pi})}{p_\phi(\boldsymbol{x}|\boldsymbol{\pi})}, \\
&= \int \mathrm{d}\boldsymbol{\pi} \int \mathrm{d}\boldsymbol{x}\; p(\boldsymbol{\pi},\boldsymbol{x})[\log q(\boldsymbol{x}|\boldsymbol{\pi}) - \log p_\phi(\boldsymbol{x}|\boldsymbol{\pi})], \\
&\geq - \int \mathrm{d}\boldsymbol{\pi} \int \mathrm{d}\boldsymbol{x}\; p(\boldsymbol{\pi},\boldsymbol{x}) \log p_\phi(\boldsymbol{x}|\boldsymbol{\pi}), \\
&\approx -\frac{1}{N}\sum_{i=1}^{N} \log p_\phi(\boldsymbol{x}_i|\boldsymbol{\pi}_i), \qquad (1)
\end{aligned}$$

where $q(\boldsymbol{x}|\boldsymbol{\pi})$ is the unknown likelihood from which the simulations $\boldsymbol{x}$ are drawn. This applies similarly for an estimator of the posterior (instead of the likelihood as shown here) and is the basis of being able to estimate the likelihood or posterior directly when an analytic form is not available. If the likelihood is fit from simulations, a prior is required and the posterior is sampled via an MCMC given some measurement. This is implemented within the code.

An ensemble of density estimators (with parameters - e.g. the weights and biases of the networks - denoted by $\{\phi_0, ..., \phi_J\}$) has a likelihood which is written as

$$p_{\text{ensemble}}(\boldsymbol{\xi}|\boldsymbol{\pi}) = \sum_{j=1}^{J} \alpha_j p_{\phi_j}(\hat{\boldsymbol{\xi}}|\boldsymbol{\pi})$$

where

$$\alpha_i = \frac{\exp(p_{\phi_i}(\hat{\boldsymbol{\xi}}|\boldsymbol{\pi}))}{\sum_{j=1}^{J} \exp(p_{\phi_j}(\hat{\boldsymbol{\xi}}|\boldsymbol{\pi}))}$$

are the weights of each density estimator in the ensemble. This ensemble likelihood can be easily sampled with an MCMC sampler. In Figure 1 we show an example posterior from applying SBI, with our code, using two compression methods separately.

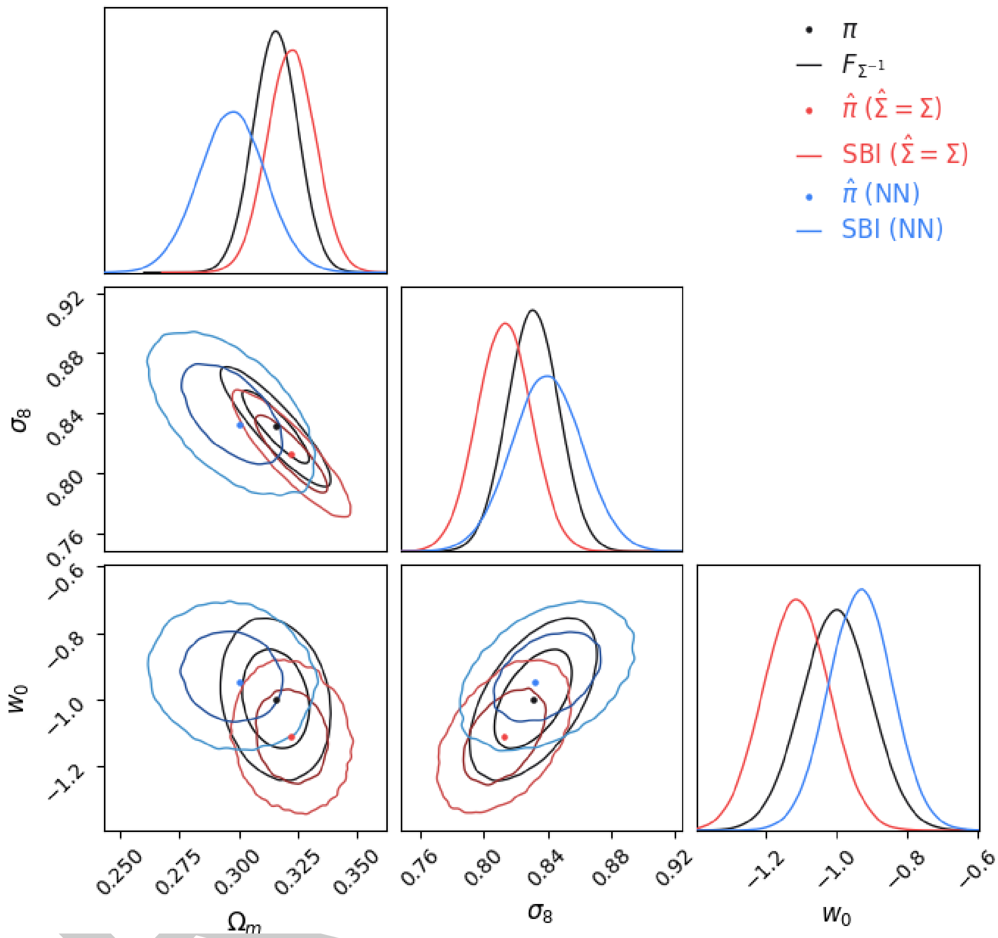

Figure 1: An example of posteriors derived with `sbiax`. We fit a ensemble of two continuous normalising flows to a set of simulations of cosmic shear two-point functions. The expectation $\xi[\pi]$ is linearised with respect to π and a theoretical data covariance model Σ allows for easy sampling of many simulations - an ideal test arena for SBI methods. We derive two posteriors, from separate experiments, where a linear (red) or neural network compression (blue) is used. In black, the true analytic posterior is shown. Note that for a finite set of simulations the blue posterior will not overlap completely with the black and red posteriors - we explore this effect upon the posteriors from SBI methods, due to an unknown data covariance, in Homer et al. (2024).

Acknowledgements

We thank the developers of the packages `jax` (Bradbury et al., 2018), `blackjax` (Cabezas et al., 2024),`optax` (DeepMind et al., 2020), `equinox` (Kidger & Garcia, 2021), `diffrax` (Kidger, 2022) and `flowjax` (Ward, 2024) for their work and for making their code available to the community.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2623–2631.

Alsing, J., Charnock, T., Feeney, S., & Wandelt, B. (2019). Fast likelihood-free cosmology with neural density estimators and active learning. *Monthly Notices of the Royal Astronomical Society*. https://doi.org/10.1093/mnras/stz1960

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 0.3.13). http://github.com/jax-ml/jax

Cabezas, A., Corenflos, A., Lao, J., & Louf, R. (2024). *BlackJAX: Composable Bayesian inference in JAX*. https://arxiv.org/abs/2402.10797

Cranmer, K., Brehmer, J., & Louppe, G. (2020). The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, *117*(48), 30055–30062. https://doi.org/10.1073/pnas.1912789117

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

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., & Duvenaud, D. (2018). *FFJORD: Free-form continuous dynamics for scalable reversible generative models*. https://arxiv.org/abs/1810.01367

Greenberg, D. S., Nonnenmacher, M., & Macke, J. H. (2019). *Automatic posterior transformation for likelihood-free inference*. https://arxiv.org/abs/1905.07488

Homer, J., Friedrich, O., & Gruen, D. (2024). *Simulation-based inference has it's own dodelson-schneider effect (but it knows it does)*. https://arxiv.org/abs/0000.00000

Kidger, P. (2022). *On neural differential equations*. https://arxiv.org/abs/2202.02435

Kidger, P., & Garcia, C. (2021). Equinox: Neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming Workshop at Neural Information Processing Systems 2021*.

Lipman, Y., Chen, R. T. Q., Ben-Hamu, H., Nickel, M., & Le, M. (2023). *Flow matching for generative modeling*. https://arxiv.org/abs/2210.02747

Papamakarios, G. (2019). *Neural density estimation and likelihood-free inference*. https://arxiv.org/abs/1910.13233

Papamakarios, G., Pavlakou, T., & Murray, I. (2018). *Masked autoregressive flow for density estimation*. https://arxiv.org/abs/1705.07057

Ward, D. (2024). *FlowJAX: Distributions and normalizing flows in jax* (Version 16.0.0). https://doi.org/10.5281/zenodo.10402073