Bioinformatics, 38(6), 2022, 1770–1772
https://doi.org/10.1093/bioinformatics/btac004
Advance Access Publication Date: 5 January 2022
Applications Note

Systems biology

Implementation of a practical Markov chain Monte Carlo sampling algorithm in PyBioNetFit

Jacob Neumann[1], Yen Ting Lin[2], Abhishek Mallela[3], Ely F. Miller[1], Joshua Colvin[1], Abell T. Duprat[1], Ye Chen[4], William S. Hlavacek[5,*] and Richard G. Posner[1,*]

[1]Department of Biological Sciences, Northern Arizona University, Flagstaff, AZ 86011, USA, [2]Information Sciences Group, Computer, Computational, and Statistical Sciences Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA, [3]Department of Mathematics, University of California, Davis, CA 95616, USA, [4]Department of Mathematics and Statistics, Northern Arizona University, Flagstaff, AZ 86011, USA and [5]Theoretical Biology and Biophysics Group, Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

*To whom correspondence should be addressed.
Associate Editor: Christina Kendziorski

Received on September 21, 2021; revised on November 30, 2021; editorial decision on December 31, 2021; accepted on January 3, 2022

Abstract

Summary: Bayesian inference in biological modeling commonly relies on Markov chain Monte Carlo (MCMC) sampling of a multidimensional and non-Gaussian posterior distribution that is not analytically tractable. Here, we present the implementation of a practical MCMC method in the open-source software package PyBioNetFit (PyBNF), which is designed to support parameterization of mathematical models for biological systems. The new MCMC method, am, incorporates an adaptive move proposal distribution. For warm starts, sampling can be initiated at a specified location in parameter space and with a multivariate Gaussian proposal distribution defined initially by a specified covariance matrix. Multiple chains can be generated in parallel using a computer cluster. We demonstrate that am can be used to successfully solve real-world Bayesian inference problems, including forecasting of new Coronavirus Disease 2019 case detection with Bayesian quantification of forecast uncertainty.

Availability and implementation: PyBNF version 1.1.9, the first stable release with am, is available at PyPI and can be installed using the pip package-management system on platforms that have a working installation of Python 3. PyBNF relies on libRoadRunner and BioNetGen for simulations (e.g. numerical integration of ordinary differential equations defined in SBML or BNGL files) and Dask.Distributed for task scheduling on Linux computer clusters. The Python source code can be freely downloaded/cloned from GitHub and used and modified under terms of the BSD-3 license (https://github.com/lanl/pybnf). Online documentation covering installation/usage is available (https://pybnf.readthedocs.io/en/latest/). A tutorial video is available on YouTube (https://www.youtube.com/watch?v=2aRqpqFOiS4&t=63s).

Contact: wish@lanl.gov or richard.posner@nau.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Given a model structure, a dataset and a prior distribution for model parameter values, Bayesian inference (Gelman *et al.*, 2014) is a dataset-dependent transformation from the prior distribution, which encodes pre-existing knowledge (external to the dataset of interest) about parameter values, to a posterior distribution, which probabilistically quantifies new data-informed parameter estimates. The posterior is typically characterized through Markov chain Monte Carlo (MCMC) sampling (Andrieu *et al.*, 2003). The MCMC sampling problems that arise in biological modeling applications of Bayesian inference are often challenging, because the posterior is typically far from Gaussian and multidimensional. A major benefit of Bayesian inference is the ability to quantify uncertainties in parameter estimates (in terms of the parameter posterior), and also in predictions, which is accomplished by performing an array of simulations based on different samples from the parameter posterior. Thus, practical MCMC sampling methods are important for assessing the reliability of predictions obtained from data-driven model parameterizations.

Few software tools provide implementations of MCMC sampling methods that enable fast setup of Bayesian inference jobs through compatibility with standardized model-specification formats used in biological modeling, such as SBML (Keating *et al.*, 2020), BNGL (Faeder *et al.*, 2009) and PySB (Lopez *et al.*, 2013). Examples of such software tools include BayesSB (Eydgahi *et al.*, 2013), PyDREAM (Shockley *et al.*, 2018), PTEMPEST (Gupta *et al.*, 2018) and PyBioNetFit (PyBNF) (Thomas *et al.*, 2016; Mitra *et al.*, 2019).

In recent work (Lin *et al.*, 2021), we used an adaptive MCMC sampling method described by Andrieu and Thoms (2008) to make daily 7-day ahead forecasts of newly detected Coronavirus Disease 2019 (COVID-19) cases with Bayesian quantification of forecast uncertainties. In this work, we found the sampling method to be easy to use and efficient. Thus, we implemented this method in PyBNF, a general-purpose software package designed to support parameterization of biological models. We also added various other features, such as support for a negative binomial likelihood function NB(r, p) wherein the hyperparameter r may be specified before inference or jointly inferred with model parameters. We evaluated new PyBNF features by solving an array of test problems and comparing the results against independently generated results.

2 Methods and implementation

PyBNF's new MCMC sampler, am, is based on the pseudocode labeled 'Algorithm 4' in Andrieu and Thoms (2008). See also Supplementary Methods. The am method is adaptive, meaning that the covariance matrix of the multivariate Gaussian proposal distribution (i.e. the distribution used to stochastically generate move proposals), is learned/optimized on-the-fly during sampling. Both the starting point for sampling and the algorithmic parameters of the proposal distribution are initialized on the basis of user-supplied inputs. Warm starts and continuation are supported. Multiple chains may be generated in an embarrassingly parallel fashion on a computer cluster and combined to decrease the wall-clock time required to complete an inference job. If this approach is used, each chain should be evaluated to check for efficient sampling as recommended in Supplementary Methods. Usage of the am method is fully explained in the online PyBNF documentation.

Inference job setup files for test problems are included in the latest PyBNF distribution. The files are also provided as Supplementary Files S1–S4, which are ZIP archives. As described in Mitra *et al.* (2019), inference job setup requires a configuration file (marked by a CONF filename extension), a model-definition file compatible with either BioNetGen (Harris *et al.*, 2016) or libRoadRunner (Somogyi *et al.*, 2015) (marked by a BNGL or XML filename extension), and one or more files containing data in a tabular format (marked by an EXP filename extension). PyBNF is compatible with models available in BNGL (Faeder *et al.*, 2009) and SBML (Keating *et al.*, 2020) formats. BNGL and SBML files can be generated by diverse tools, including BioNetGen (Harris *et al.*, 2016), PySB (Lopez *et al.*, 2013) and Tellurium (Choi *et al.*, 2018).

3 Results

To assess the correctness and practicality of the am method, we used it to solve a series of increasingly challenging test problems. We evaluated am relative to mh, a sampler implemented in PyBNF version 1.01 (Mitra *et al.*, 2019) that uses a fixed move proposal distribution, and against PyBNF-independent problem-specific solutions.

We started with a linear regression problem involving synthetic data (Supplementary File S1 and Supplementary Methods). For this problem, an analytical expression for the posterior exists. PyBNF's am method was able to reconstruct the analytical posterior nearly exactly (Supplementary Fig. S1), whereas mh failed with the same computational budget.

We next considered a non-linear regression problem: inferring the values of four parameters in a two-phase exponential decay model for viral dynamics under therapy (Ho *et al.*, 1995; Perelson *et al.*, 1996; Supplementary File S2 and Supplementary Methods). Results generated using PyBNF's am method were found to be consistent with results generated using problem-specific code (Supplementary Figs S2–S4).

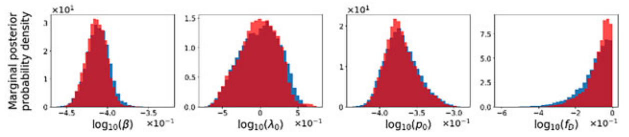

Fig. 1. A comparison of selected marginal posteriors was obtained using PyBNF's am method (blue) and the problem-specific code of Lin *et al.* (2021; light red). Dark red indicates overlap. The inference problem is that considered in Supplementary Figure S7, which is related to a forecast of COVID-19 incidence for the metropolitan statistical area encompassing Los Angeles, CA. Nomenclature is the same as that of Lin *et al.* (2021)

Using a previously established benchmark problem (Harmon *et al.*, 2017; Mitra *et al.*, 2019; Supplementary File S3 and Supplementary Methods), we evaluated the efficiency of am relative to mh. Results obtained using the two methods are consistent (Supplementary Fig. S5). For each of the 16 adjustable parameters in the benchmark problem, we calculated the effective sample size (ESS) for an equal-length chain of parameter values obtained using either am or mh (Supplementary Methods and Supplementary Table S1). ESS for am was typically larger than for mh, indicating greater efficiency. The ratio $\text{ESS}_{\text{am}}/\text{ESS}_{\text{mh}}$ ranged from 3.03 to 27.7, with a mean (median) of 11.5 (10.5).

Finally, we used PyBNF's am method to reproduce inferences performed in the epidemiological forecasting study of Lin *et al.* (2021; Supplementary File S4 and Supplementary Methods). As illustrated in Figure 1, marginal posteriors found using the problem-specific code of Lin *et al.* (2021) and am are indistinguishable. Supplementary Figures S6–S20 show the full results of our comparison of am against the code of Lin *et al.* (2021).

4 Conclusions

We find that the new am method available in PyBNF version 1.1.9 is significantly more efficient than the mh method available in earlier versions of PyBNF. We also find that the adaptive sampler is practical, in that it can be used to solve real-world inference problems, including challenging inference problems that arose in a recent COVID-19 forecasting effort (Lin *et al.*, 2021; Fig. 1 and Supplementary Figs S6–S20).

Funding

This work was supported by NIH/NIGMS grant [R01GM111510] and the LDRD program at Los Alamos National Laboratory.

Conflict of Interest: none declared.

References

Andrieu,C. *et al.* (2003) An introduction to MCMC for machine learning. *Mach. Learn.*, **50**, 5–43.

Andrieu,C. and Thoms,J. (2008) A tutorial on adaptive MCMC. *Stat. Comput.*, **18**, 343–373.

Choi,K. *et al.* (2018) Tellurium: an extensible python-based modeling environment for systems and synthetic biology. *Biosystems*, **171**, 74–79.

Eydgahi,H. *et al.* (2013) Properties of cell death models calibrated and compared using Bayesian approaches. *Mol. Syst. Biol.*, **9**, 644.

Faeder,J.R. *et al.* (2009) Rule-based modeling of biochemical systems with BioNetGen. *Methods Mol. Biol.*, **500**, 113–167.

Gelman,A. *et al.* (2014) *Bayesian Data Analysis*, 3rd edn. CRC Press, Boca Raton, FL.

Gupta,S. *et al.* (2018) Evaluation of parallel tempering to accelerate Bayesian parameter estimation in systems biology. In: *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based*

Processing (PDP), Institute of Electrical and Electronics Engineers (IEEE), Piscataway, New Jersey, USA, pp. 690–697.

Harmon,B. *et al.* (2017) Timescale separation of positive and negative signaling creates history-dependent responses to IgE receptor stimulation. *Sci. Rep.*, **7**, 15586.

Harris,L.A. *et al.* (2016) BioNetGen 2.2: advances in rule-based modeling. *Bioinformatics*, **32**, 3366–3368.

Ho,D.D. *et al.* (1995) Rapid turnover of plasma virions and CD4 lymphocytes in HIV-1 infection. *Nature*, **373**, 123–126.

Keating,S.M. *et al.*; SBML Level 3 Community members (2020) SBML Level 3: an extensible format for the exchange and reuse of biological models. *Mol. Syst. Biol.*, **16**, e9110.

Lin,Y.T. *et al.* (2021) Daily forecasting of regional epidemics of Coronavirus Disease with Bayesian uncertainty quantification, United States. *Emerg. Inf. Dis.*, **27**, 767–778.

Lopez,C.F. *et al.* (2013) Programming biological models in Python using PySB. *Mol. Syst. Biol.*, **9**, 646.

Mitra,E.D. *et al.* (2019) PyBioNetFit and the biological property specification language. *iScience*, **19**, 1012–1036.

Perelson,A.S. *et al.* (1996) HIV-1 dynamics in vivo: virion clearance rate, infected cell life-span, and viral generation time. *Science*, **271**, 1582–1586.

Shockley,E.M. *et al.* (2018) PyDREAM: high-dimensional parameter inference for biological models in python. *Bioinformatics*, **34**, 695–697.

Somogyi,E.T. *et al.* (2015) LibRoadRunner: a high performance SBML simulation and analysis library. *Bioinformatics*, **31**, 3315–3321.

Thomas,B. *et al.* (2016) BioNetFit: a fitting tool compatible with BioNetGen, NFsim and distributed computing environments. *Bioinformatics*, **32**, 798–800.