SOFTWARE TOOL ARTICLE

PYLFIRE: Python implementation of likelihood-free inference by ratio estimation [version 1; peer review: 2 approved]

Jan Kokko [1], Ulpu Remes [1], Owen Thomas[2], Henri Pesonen[2], Jukka Corander [1-3]

[1]Department of Mathematics and Statistics, University of Helsinki, Helsinki, Finland
[2]Department of Biostatistics, University of Oslo, Oslo, Norway
[3]Parasites and Microbes, Wellcome Trust Sanger Institute, Hinxton, UK

v1 **First published:** 10 Dec 2019, **4**:197 (https://doi.org/10.12688/wellcomeopenres.15583.1)
Latest published: 10 Dec 2019, **4**:197 (https://doi.org/10.12688/wellcomeopenres.15583.1)

Abstract

Likelihood-free inference for simulator-based models is an emerging methodological branch of statistics which has attracted considerable attention in applications across diverse fields such as population genetics, astronomy and economics. Recently, the power of statistical classifiers has been harnessed in likelihood-free inference to obtain either point estimates or even posterior distributions of model parameters. Here we introduce PYLFIRE, an open-source Python implementation of the inference method LFIRE (likelihood-free inference by ratio estimation) that uses penalised logistic regression. PYLFIRE is made available as part of the general ELFI inference software http://elfi.ai to benefit both the user and developer communities for likelihood-free inference.

Keywords

density-ratio estimation, likelihood-free inference, logistic regression, summary statistics selection

Open Peer Review

Reviewer Status ✓ ✓

	Invited Reviewers	
	1	**2**
version 1 10 Dec 2019	✓ report	✓ report

1 **Samuel Wiqvist**, Lund University, Lund, Sweden

2 **Guilherme Rodrigues**, University of Brasília, Brasilia, Brazil

Any reports and responses or comments on the article can be found at the end of the article.

Corresponding author: Jan Kokko (jan.kokko@helsinki.fi)

Author roles: Kokko J: Conceptualization, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Remes U**: Conceptualization, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Thomas O**: Conceptualization, Writing – Review & Editing; **Pesonen H**: Conceptualization, Writing – Review & Editing; **Corander J**: Conceptualization, Supervision, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: This work was supported by the Wellcome Trust [206194]. This work was also supported by the Academy of Finland [316602] (JK, UR), and the European Research Council [742158] (OT, HP, JC).
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

How to cite this article: Kokko J, Remes U, Thomas O *et al.* **PYLFIRE: Python implementation of likelihood-free inference by ratio estimation [version 1; peer review: 2 approved]** Wellcome Open Research 2019, **4**:197 (https://doi.org/10.12688/wellcomeopenres.15583.1)

First published: 10 Dec 2019, **4**:197 (https://doi.org/10.12688/wellcomeopenres.15583.1)

Introduction

Computer-simulator-based models are becoming increasingly popular across a wide spectrum of scientific applications as they allow a more flexible framework for encoding expert knowledge than typical statistical model families. A simulator model must nevertheless under most circumstances be carefully tuned to produce realistic outcomes when compared with observed data from some process that the simulator is trying to mimic. Likelihood-free inference deals with the need to estimate parameters and quantify uncertainties in parameter values in the light of the observations x when using simulator models. The most common approaches in likelihood-free inference include approximate Bayesian computation (ABC)[1,2] and synthetic likelihood (SL)[3,4]. In a general statistical setting, Bayesian inference combines a probability model for the observations x with the parameter(s) θ and a prior distribution for the parameters $p(\theta)$ to define the posterior distribution over parameters θ as

$$p(\theta \mid x) = \frac{p(\theta)p(x \mid \theta)}{p(x)}, \tag{1}$$

where $p(x|\theta)$ denotes the likelihood function and $p(x)$ denotes the marginal likelihood defined as $p(x) = \int p(\theta)p(x|\theta)d\theta$. The present work considers posterior estimation when the likelihood function is not available, but assuming that synthetic data can be be generated from the model given a configuration of its parameters. Posterior distribution or at least some summaries for it must then be obtained with likelihood-free inference methods, such as ABC or SL. The methods typically use summary statistics $\psi(x)$ that are engineered to capture the relevant information about the model parameters present in the real or simulated observations x.

Recent advances in likelihood-free methods include likelihood-free inference by ratio estimation (LFIRE)[5]. LFIRE converts the posterior estimation problem into a density-ratio estimation problem that can be solved with logistic regression[6]. Assume that the dataset $\{x_n^\theta\}$ includes N observations simulated using particular parameter values θ and that $\{x_n^m\}$ includes observations simulated from the marginal data distribution, which is obtained by averaging the forward simulation process over random parameter values sampled from the prior distribution $\theta \sim p(\theta)$. Logistic regression is then used in LFIRE to model the log-ratio

$$h(x) = \log \frac{p(x \in \{x_n^\theta\})}{p(x \in \{x_n^m\})} \tag{2}$$

which approximates the log-ratio between the likelihood and marginal likelihood functions evaluated at θ. Thomas *et al.*[5] propose modelling the log-ratio as a sparse linear combination of summary statistics $\psi_i(x)$ calculated from the observations x:

$$h(x) = \beta_0 + \sum_i \beta_i \psi_i(x) = \beta^\top \psi(x), \tag{3}$$

where β denotes the linear model parameters and $\psi(x)$ contains the summary statistics and a constant term. Estimation of the posterior is then based on the idea that we: (1) find the linear model parameters $\hat{\beta}(\theta)$ that minimise the $l1$-penalised logistic loss function evaluated over observation sets $\{x_n^\theta\}$ and $\{x_n^m\}$, and (2), use the estimated log-ratio model to approximate the posterior density with

$$\hat{p}(\theta \mid x) = p(\theta) \exp(\hat{\beta}(\theta)^\top \psi(x)). \tag{4}$$

Summary statistics that contribute in the posterior estimation are selected automatically since $l1$ penalisation promotes sparse solutions where the coefficients are forced to zero when the corresponding summary statistic is deemed irrelevant for the prediction: $\beta_i = 0$. Since the model for the approximate posterior is estimated based on synthetic observation sets $\{x_n^\theta\}$ and $\{x_n^m\}$, we can control the dataset size to ensure accurate parameter estimation.

To summarise, LFIRE uses lasso logistic regression to approximate the intractable likelihood function and to achieve data-driven selection of summary statistics in likelihood-free inference. Related works include sparse precision matrix estimation in synthetic likelihood approaches[7] and semi-parametric synthetic likelihood[8]. LFIRE implementations are currently available in MATLAB[5] and in ABCpy (Python)[9] and the related synthetic likelihood methods in BSL (R)[10], while most other likelihood-free inference tools are focussed on ABC and related summary statistic selection methods. General-purpose tools most relevant to the current contribution are reviewed in 11 and available summary statistics selection methods in 12.

The present work introduces a new LFIRE implementation that is compatible with models constructed with the ELFI software[11]. ELFI is a general-purpose likelihood-free inference software that provides tools for modelling inference problems and includes various ABC approaches. Our PYLFIRE implementation introduced here extends ELFI by adding LFIRE to its pool of available inference methods. PYLFIRE implementation is discussed in closer detail in the next section, after which we provide operational instructions and demonstrate the workflow for estimating a posterior distribution with the software.

Methods

Implementation

PYLFIRE generates a marginal or prior predictive observation set $\{x_n^m\}$ by sampling N configurations from the prior distribution $p(\theta)$ and conditional observation sets $\{x_n^\theta\}$ by sampling corresponding observations from the observation model $p(x|\theta)$. The observation sets $\{x_n^\theta\}$ are generated with parameter combinations θ that indicate the locations in parameter space where lasso logistic regression is used to calculate an approximate log-ratio and approximate posterior based on $\{x_n^\theta\}$ and $\{x_n^m\}$. Hence the main computational tasks are dataset generation and fitting the logistic regression.

PYLFIRE constructs the simulated datasets with ELFI 0.7.4[11] and estimates the sparse logistic regression model with glmnet 2.1.1[13]. ELFI models inference problems as networks that are used in PYLFIRE to generate observations with random parameters that follow the user-specified prior distribution or with fixed parameter values. Logistic regression parameters are estimated with the glmnet implementation that utilises Fortran subroutines for fast execution. Estimation is based on cyclical coordinate descent to minimise the penalised loss function with respect to regression parameters and cross-validation to optimise the penalisation level. When multiple computation cores are available, PYLFIRE can parallellise dataset construction or cross-validation.

Operation

PYLFIRE requires Python 3.6.0 (or a later version) and a Fortran compiler. However we also provide a Docker container image to run PYLFIRE with the requirements pre-installed. The package and installation instructions are available in ELFI zoo https://github.com/elfi-dev/zoo/tree/master/pylfire. PYLFIRE is provided with a makefile and installation options as follows. To install PYLFIRE in an environment that has Python 3.6 and a Fortran compiler, run installation:

```
make install
```

We then recommend testing the PYLFIRE framework with:

```
make test
```

The alternative is to build and run the PYLFIRE Docker image:

```
make docker-build
make docker-run
```

This requires that Docker is installed, but avoids possible problems with the Fortran compiler. When PYLFIRE is installed, it is available with:

```
import pylfire
```

Running posterior inference with LFIRE implemented in the PYLFIRE package then includes (1) ELFI model construction and (2) running LFIRE to estimate posterior probabilities at predetermined parameter combinations. ELFI model construction means that the user adds their parameter priors, simulator, and summarisation rules into a network structure. While model construction does not require observed data, observations can be added in the model. The process is demonstrated in ELFI documentation and tutorials: https://elfi.readthedocs.io/.

ELFI model provides PYLFIRE means to generate observations from the marginal and conditional distributions. In addition to the network model, the user must determine the parameter combinations where approximate posterior is evaluated and the dataset size used in logistic regression. The observed data x must also be added in the network model, and the model, parameter combinations, and dataset size are then used to initialise an LFIRE instance that calculates the approximate posterior probabilities. The process is demonstrated in the next section.

Use case

We demonstrate how ELFI and PYLFIRE can be used to estimate the posterior distribution over model parameters in a lag-one autoregressive model with conditional heteroscedasticity (ARCH(1)). The model describes dependencies between observations in a time series as

$$y^{(t)} = \theta_1 y^{(t-1)} + e^{(t)} \tag{5}$$

$$e^{(t)} = \xi^{(t)} \sqrt{0.2 + \theta_2 (e^{(t-1)})^2}, \tag{6}$$

where $y^{(0)} = 0$, and $e^{(0)}$ and $\xi^{(t)}$ are independent standard normal random variables. The time series used as observed data in the current example is simulated with parameters $(\theta_1, \theta_2) = (0.3, 0.7)$ and has $T = 100$ observations. The simulator is available in the PYLFIRE package:

```
from pylfire.models import arch
```

Dependencies between parameters, observations, and summary statistics are described in a predetermined ELFI model that is loaded with:

```
m = arch.get_model()
```

Here the ARCH(1) simulator parameters are associated with prior distributions $\theta_1 \sim \mathcal{U}(-1, 1)$ and $\theta_2 \sim \mathcal{U}(0, 1)$. In our codes we denote the parameters θ_1 and θ_2 as t1 and t2, respectively. The parameters are mapped into time series observations with the simulator, and observations are in turn reduced into summaries that include the time series mean, variance, autocorrelations, and pairwise combinations between the autocorrelations, as described in previous work[5]. Figure 1 illustrates the network structure of the ARCH(1) ELFI model.

LFIRE determines an approximate posterior distribution over parameter values based on the summaries calculated from an observed time series. At a minimum, the method requires three inputs: an ELFI model with the observed data, candidate parameter combinations, and the dataset size to be used in logistic regression. Parameter combinations must be given as a two-dimensional numpy array where columns correspond to the individual parameters and rows correspond to the parameter combinations to be evaluated. In the current example we estimate posterior probabilities on a 100 × 100 grid over the [–1, 1] × [0,1] parameter space:

```
import numpy as np

n = 100
t1 = np.linspace(-1, 1, n)
t2 = np.linspace(0, 1, n)
tt1, tt2 = np.meshgrid(t1, t2, indexing='ij')
params_grid = np.c_[tt1.flatten(), tt2.flatten()]
```

and the dataset size used in logistic regression must be given as an integer:

```
batch_size = 1000
```

LFIRE is then initialised with:

```
lfire_method = pylfire.LFIRE(model=m, params_grid=params_grid, batch_size=batch_
size)
```

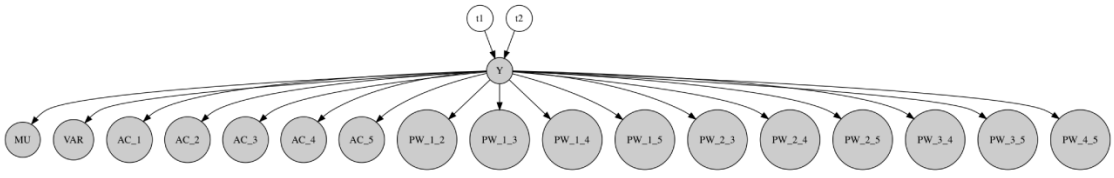

Figure 1. Network structure of the ARCH(1) model.

LFIRE can additionally take precomputed marginal data and custom logistic regression parameters as optional inputs, and allows the user to control whether cross-validation or dataset generation is run in parallel. By default LFIRE generates the marginal data when initialised, uses lasso logistic regression, and runs cross-validation in parallel.

After the LFIRE method is initialised, one can run inference and extract results with:

```
lfire_res = lfire_method.infer()
lfire_res.summary()

Method: LFIRE
Number of simulations: 10000000
MAP estimates: t1: 0.434, t2: 0.515
Posterior means: t1: 0.388, t2: 0.654
```

PYLFIRE also provides two visualisation methods: one for plotting marginal posterior densities and another for plotting pairwise posterior densities:

```
lfire_res.plot_marginals()
lfire_res.plot_pairs()
```

Figure 2 visualises the marginal posterior distributions and Figure 3 visualises the pairwise marginal posterior distributions. PYLFIRE also records the logistic regression parameters estimated at each candidate location so that users can examine how summary statistics were weighted in posterior estimation and determine whether automatic selection focussed on certain summaries. All results are stored in a dictionary and can be extracted with:

```
lfire_res.results
```

Finally, LFIRE incorporates automatic summary statistic selection to make posterior estimation robust to irrelevant summary statistics. In this experiment the summary statistics in ARCH(1) model are augmented with 17 white-noise variables:

```
m_noisy = arch.get_model(noise=17)

lfire_method_noisy = pylfire.LFIRE(model=m_noisy, params_grid=params_grid,
batch_size=batch_size)

lfire_res_noisy = lfire_method_noisy.infer()
lfire_res_noisy.summary()

Method: LFIRE
Number of simulations: 10000000
MAP estimates: t1: 0.475, t2: 0.556
Posterior means: t1: 0.394, t2: 0.64
```

Comparison to previous results confirms that the point estimates and posterior distribution remain about the same despite irrelevant summary statistics. The estimated posterior distribution is visualised in Figure 4.

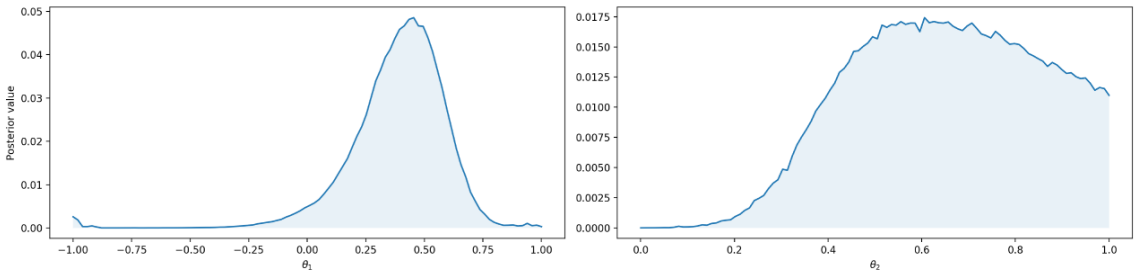

Figure 2. Marginal posterior distributions for parameters θ_1 and θ_2.

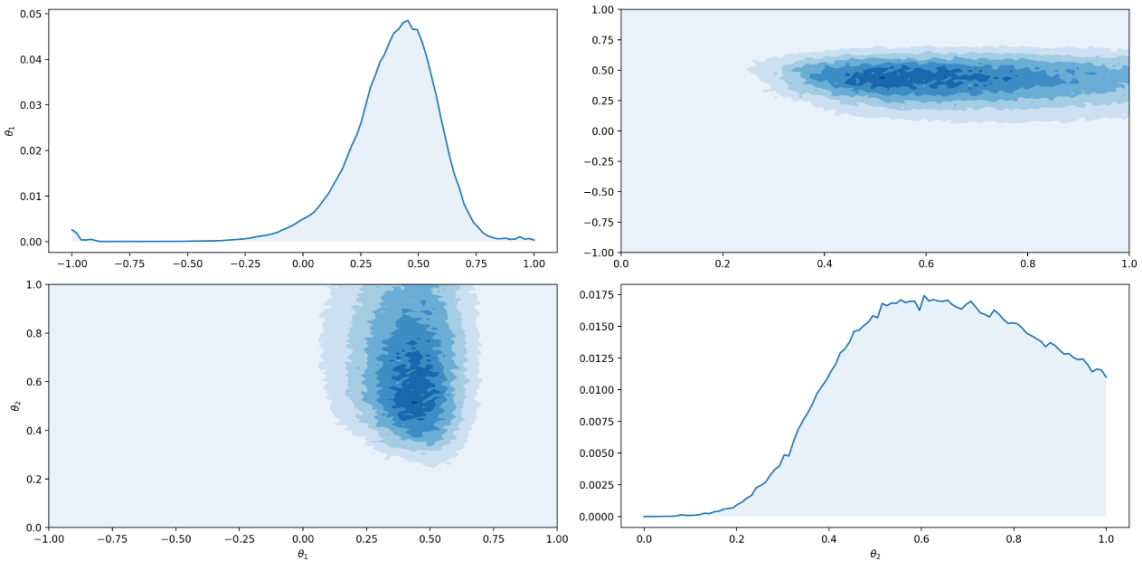

Figure 3. Posterior and marginal distributions for parameters θ_1 and θ_2.

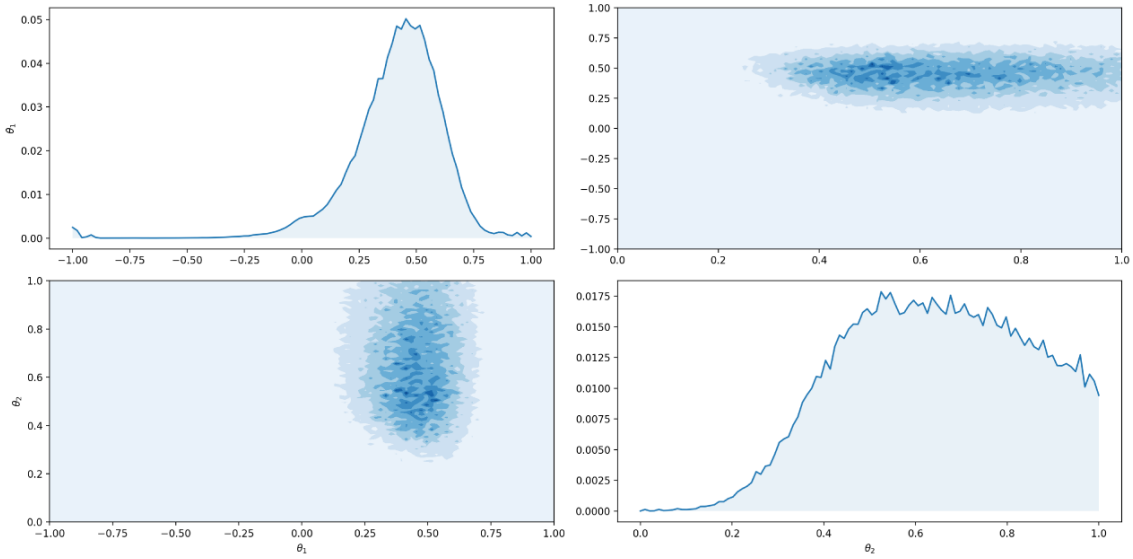

Figure 4. Posterior and marginal distributions for parameters θ_1 and θ_2 when model includes irrelevant summary statistics.

Summary

We have introduced PYLFIRE, an open-source Python package running on ELFI and implementing the ratio-estimation-based LFIRE method for likelihood-free inference with automatic summary statistic selection is implemented. PYLFIRE seeks to minimise the computation time in LFIRE with parallelisation and using the external glmnet package[13] where key components are written in Fortran. PYLFIRE uses glmnet to fit lasso logistic regression. For convenience, PYLFIRE provides summarised inference results and two built-in plotting methods for visualising the estimated posterior distribution.

Data availability

Underlying data

All data underlying the results are available as part of the article and no additional source data are required.

Software availability

1. Source code available from: https://github.com/elfi-dev/zoo/tree/master/pylfire
2. Archived source code as at time of publication: https://doi.org/10.5281/zenodo.3533332[14].
3. License: BSD 3-Clause

References

1. Sisson SA, Fan Y, Beaumont MA: **Handbook of approximate Bayesian computation.** CRC Press. 2019.
Reference Source
2. Lintusaari J, Gutmann MU, Dutta R, *et al.*: **Fundamentals and Recent Developments in Approximate Bayesian Computation.** *Syst Biol.* 2017; **66**(1): e66–e82.
PubMed Abstract | Publisher Full Text | Free Full Text
3. Wood SN: **Statistical inference for noisy nonlinear ecological dynamic systems.** *Nature.* 2010; **466**(7310): 1102–1104.
PubMed Abstract | Publisher Full Text
4. Price LF, Drovandi CC, Lee A, *et al.*: **Bayesian synthetic likelihood.** *J Comput Graph Stat.* 2018; **27**(1): 1–11.
Publisher Full Text
5. Thomas O, Dutta R, Corander J, *et al.*: **Likelihood-free inference by ratio estimation.** arXiv:1611.10242. 2016.
Reference Source
6. Sugiyama M, Suzuki T, Kanamori T: **Density Ratio Estimation in Machine Learning.** Cambridge University Press. 2012.
Publisher Full Text
7. An Z, South LF, Nott DJ, *et al.*: **Accelerating Bayesian synthetic likelihood with the graphical lasso.** *J Comput Graph Stat.* 2019; **28**(2): 471–475.
Publisher Full Text
8. An Z, Nott DJ, Drovandi C: **Robust Bayesian synthetic likelihood via a semi-parametric approach.** arXiv:1809.05800. 2018.
Reference Source
9. Dutta R, Schoengens M, Ummadisingu A, *et al.*: **ABCpy: A high-performance computing perspective to approximate Bayesian computation.** arXiv:1711.04694. 2017.
Reference Source
10. An Z, South LF, Drovandi C: BSL: **BSL: An R package for efficient parameter estimation for simulation-based models via Bayesian synthetic likelihood.** arXiv:1907.10940. 2019.
Reference Source
11. Lintusaari J, Vuollekoski H, Kangasrääsiö A, *et al.*: **ELFI: Engine for Likelihood-Free Inference.** *J Mach Learn Res.* 2018; **19**(16): 1–7.
Reference Source
12. Nunes MA, Prangle D: **abctools: an R package for tuning approximate Bayesian computation analyses.** *R Journal.* 2016; **7**(2): 189–205.
Reference Source
13. Friedman J, Hastie T, Tibshirani R: **Regularization Paths for Generalized Linear Models via Coordinate Descent.** *J Stat Softw.* 2010; **33**(1): 1–22.
PubMed Abstract | Publisher Full Text | Free Full Text
14. Kokko J, Remes U, Thomas O, *et al.*: **Pylfire.** 2019.
http://www.doi.org/10.5281/zenodo.3533332

Open Peer Review

Current Peer Review Status: ✔ ✔

Version 1

Reviewer Report 24 February 2020

https://doi.org/10.21956/wellcomeopenres.17064.r37720

✔ **Guilherme Rodrigues**
Department of Statistics, University of Brasília, Brasilia, Brazil

Summary of the paper

In modern statistical analysis, the adoption of complex probabilistic models frequently results in likelihood functions that are computationally costly to evaluate (*intractable*). A vast collection of methods has been engineered to enable statistical inference in such situations. In particular, Approximate Bayesian Computation (ABC)[1] and Synthetic likelihood (SL)[2] are well established classes of likelihood-free algorithms. As an alternative approach, Thomas *et al.* (2016)[3] proposed a novel technique, named LFIRE (likelihood-free inference by ratio estimation), which converts the problem of estimating the posterior distribution as a problem of modeling (by lasso logistic regression fitted over *synthetic/pseudo* samples) the ratio between the data generating distribution and the marginal distribution. A convenient byproduct of such formulation is the resulting semi-automatic selection of summary statistics implied by the lasso (least absolute shrinkage and selection operator) regularization.

This paper introduces PYLFIRE, an open-source Python implementation of LFIRE that provides for practitioners and developers an easy-to-operate toll for parameter estimation using LFIRE. In an introduction section, the paper describes the general problem of likelihood-free inference in the Bayesian framework and then outline how LFIRE approaches such problem. Next, in a methodological section, it elaborates on the implementation and operational aspects of the package, including details about the installation process and the support offered for parallel computing. A simple bi-dimensional autoregressive model with conditional heteroscedasticity (ARCH(1)) model is then fitted with PYLFIRE for illustrational purposes. To conclude, the paper summarizes the package main features and implementation decisions.

Major comments:
The paper is generally well-written and serves its purpose as a helpful reference source for the PYLFIRE package documentation. The package itself also provides a valuable contribution to the initiative of offering efficient open-source tools that allows immediate exploitation of some of the most advanced solutions for the likelihood-free estimation problem.

1. Albeit the focus of the paper is, understandably, the computational aspects of the package, it would be beneficial for the reader if the paper was self-contained, in the sense that all information regarding the practical use of the package was present. The description of the LFIRE method, especially, should be meaningfully extended. In this regard, I recommend the inclusion of a pseudo-algorithm that lists the inputs, outputs and the steps involved in the referred approximate posterior sampling algorithm. Moreover, throughout the introduction, a more explicit distinction between *observed* and *synthetic* data should be pursued.

2. It would also be of great use to include some guidance on how to specify the parameters of the algorithm (e.g. the number of simulated samples from the marginal distribution and from the data generating distribution) and to discuss when the method is suitable. For instance, models with a moderate number of parameters or where the prior is diffuse (as compared to the posterior) seem to be out of reach.

3. Considering that another Python routine is currently available, namely, ABCpy, a deeper comparison of the relative merits and deficiencies of each implementation seems necessary. A well-designed numerical simulation would certainly contribute to the robustness of the comparison.

Minor comments:

1. The notation used for the sample sizes is a bit confusing and could be improved. Is there a reason to use lowercase (n) and uppercase (N) letters? In addition, the number of simulated samples from the marginal distribution and from the data generating distribution are necessarily the same?

2. The term “is implemented” should be suppressed from the first paragraph of the Summary section.

3. It is not sufficiently clear how the observed summary statistics are passed on to the function when the user itself provides the simulated data sets.

4. I wonder how feasible it would be to embed the provided functionalities in an optimization process that aims to find the MAP (maximum a posterior probability). If possible, a brief discussion on how to accomplish that could extend the reach of the package when estimating the whole posterior distribution is impracticable.

5. The lasso regression is a reasonable and convenient classifier for estimating the density ratio. However, other classification algorithms could, at least in principle, be employed, either for better accuracy or for computational speed. As such, have the authors considered giving the user control of the classification process, for example, allowing them to pass on a custom python function that returns the estimated classification probabilities (instead of the reported optional “custom logistic regression parameters”)? If feasible, this extension would readily benefit from advances on the probabilistic classification area (e.g. whenever a new package for classification becomes available or in cases where the glmnet 2.1.1 package[4] underperforms).

Conclusions

The paper is well-structured, generally clear and represents a valuable addition to the practitioner’s toolbox. For the readers convenience, an extra effort should be made to ensure the paper is self-contained. A list of suggestions was included in the comments above to indicate some options that could be explored in this regard.

References

1. Sisson SA, Fan Y, Beaumont MA: Handbook of approximate Bayesian computation. *CRC Press*. 2019.

2. Price L, Drovandi C, Lee A, Nott D: Bayesian Synthetic Likelihood. *Journal of Computational and Graphical Statistics*. 2018; **27** (1): 1-11 Publisher Full Text
3. Thomas O, Dutta R, Corander J, Kaski S, Gutmann MU: Likelihood-free inference by ratio estimation. *arXiv*. 2016.
4. Friedman J, Hastie T, Tibshirani R: Regularization Paths for Generalized Linear Models via Coordinate Descent.*J Stat Softw*. 2010; **33** (1): 1-22 PubMed Abstract

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Approximate Bayesian computation; Bayesian Dynamic models.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Reviewer Report 21 January 2020

https://doi.org/10.21956/wellcomeopenres.17064.r37335

Samuel Wiqvist
Centre for Mathematical Sciences, Lund University, Lund, Sweden

Introduction to the topic

The problem that is considered in this paper is parameter inference for implicit statistical models (i.e. computer-simulator model) in the Bayesian framework. In particular, what the authors aim at is posterior estimation when the likelihood function is intractable, but data-simulation from the model is available, i.e.

what is considered is a likelihood-free inference method. The paper presents the PYLFIRE package (here), which is an open-source Python package that implements the likelihood-free inference by ratio estimation (LFIRE) method [6]. The LFIRE method is a simulation-based inference method for implicit Bayesian models, where the inference problem is transformed into a density-ration estimation problem, which is solved using Lasso logistic regression [6]. Thus, LFIRE has similarities to approximate Bayesian computation (ABC) [5] and in-particular synthetic likelihood (SL) [4]. However, in contrast to SL, LFIRE does not require any distribution assumptions on the summary statistics [6]. Another feature of the LFIRE method is that it allows for automatic selection of summary statistics by employing Lasso logistic regression [6]. This is an interesting feature since selecting proper summary statistics is one of the main challenges when using ABC in practice.

Summary of paper

The article starts by introducing the implicit statistical model (i.e., computer-simulator model); in the introduction, the LFIRE method is also briefly introduced. Then the authors explain how PYLFIRE is structured: PYLFIRE extends the ELFI software [1] and utilizes the ELFI framework, and the glmnet 2.1.1 package [2] is utilized for logistic regression modeling. The authors stress that PYLFIRE is developed with performance in mind; therefore, fast FORTRAN subroutines and parallelization are supported. Instructions on how to install PYLFIRE are provided, and a simple case study (simulation study for an ARCH(1) model) highlights the capabilities of PYLFIRE. The paper ends with a summary of the most important features of PYLFIRE.

Comments

The introduction in the paper is quite useful to the reader since it contains a short general introduction to the LFIRE method. However, this could be extended, which would make it easier for a reader not already familiar with the LFIRE method to follow the paper. An idea could be to try to include in the introduction a similar figure as to Figure 1 in [6].

The authors' reason for developing PYLFIRE could be further processed. Is PYLFIRE developed primarily to extend the capabilities of the ELFI software, to provide an easy-to-use package for LFIRE based inference, or to provide an LFIRE implementation with better computational performance than other implementations?

Another minor comment is that the authors could include a link to the Jupyter notebook (here) with the ARCH(1) example in the beginning of the *Use case* section. This would make it easier for a reader to get started using PYLFIRE.

The authors provide an associated Docker image for PYLFIRE, which is very useful since a user then can circumvent the possibly somewhat involved installation process.

PYLFIRE implements the LFIRE method [6]; however [6] appears to be still unpublished at the time of writing. It would, therefore, be useful to know if [6] is currently under review in some venue.

Regarding computational performance: The authors mention that PYLFIRE is implemented with computational performance in mind. Due to this, the authors utilize parallelization and efficient FORTRAN subroutines (via the glmnet 2.1.1 package [2]). However, in the paper, it is not made clear if these efforts of writing efficient code rendered any performance improvements compared to other implementations of the LFIRE method. The authors state that there already exists a MATLAB implementation [6] and another

Python implementation (ABCpy [1]) of the LFIRE method. A computational performance comparison with these implementations (or at least the ABCpy implementation) would be of interest, in order to investigate if PYLFIRE indeed has a better computational performance. A comparison with the other implementations could also investigate how easy the packages are to use, and possibly show the reader the potential advantages of using PYLFIRE.

Furthermore, PYLFIRE utilizes parallelization by default. However, in the paper, the authors do not investigate the potential performance gains from utilizing parallelization. Setting up such a case study is, of course, quite tricky since the results will be highly dependent on the model considered and the particular computer-system used. However, including such a case study could still be of interest since it could highlight: 1) how the user can run the PYLFIRE package in parallel, 2) some idea on what performance gains are plausible.

The only case study provided in the paper is a simulation study for the ARCH(1) model. In particular, since computational performance is of interest, would it be interesting to see how the PYLFIRE performs for a more complicated model with a higher-dimensional parameter space. Such a case study could potentially also highlight the parallel computation capabilities of PYLFIRE.

Conclusion

This is an overall clear and well-written paper that for the most parts is easy to follow. However, some parts of the paper could be extended for clarification, and the paper would be improved by including some further analyses (for details, see the *Comments* section). The paper fulfils article approval status - approved, but it could be further improved by incorporating the suggestions presented in the *Comments* section.

References

1. Dutta R, Schoengens M, Ummadisingu A, Widmer N, Onnela J, Mira A: ABCpy: A High-Performance Computing Perspective to Approximate Bayesian Computation. *arXiv*. 2017.
2. Friedman J, Hastie T, Tibshirani R: Regularization Paths for Generalized Linear Models via Coordinate Descent.*J Stat Softw*. 2010; **33** (1): 1-22 PubMed Abstract
3. Lintusaari J, Gutmann M, Dutta R, Kaski S, Corander J: Fundamentals and Recent Developments in Approximate Bayesian Computation. *Systematic Biology*. 2016. Publisher Full Text
4. Price L, Drovandi C, Lee A, Nott D: Bayesian Synthetic Likelihood. *Journal of Computational and Graphical Statistics*. 2018; **27** (1): 1-11 Publisher Full Text
5. Sisson SA, Fan Y, Beaumont MA: Handbook of approximate Bayesian computation. *CRC Press*. 2019.
6. Thomas O, Dutta R, Corander J, Kaski S, Gutman MU: Likelihood-free inference by ratio estimation. *arXiv*. 2016.

Is the rationale for developing the new software tool clearly explained?

Partly

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Likelihood-free inference methods

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.