ELFI: Engine for Likelihood-Free Inference

Antti Kangasrääsiö,
Jarno Lintusaari,
Kusti Skytén,
Marko Järvenpää,
Henri Vuollekoski
Michael Gutmann,
Aki Vehtari,
Jukka Corander,
Samuel Kaski

Abstract

We introduce an *Engine for Likelihood-Free Inference* (ELFI), a software package for approximate Bayesian inference [1] that can be used when the likelihood function is difficult to evaluate or unknown, but a generative simulator model exists.

The software is in Python, and its modular library design emphasizes both ease-of-use and expandability, allowing arbitrary user-defined simulators and implementation of new inference methods with minimal effort.

Probabilistic inference models can be represented intuitively as graphs, and users can execute the inference in a computational environment best suited for their needs, from single laptops to cluster computers.

The whole inference pipeline is automatically parallelized, and intermediate results may be stored to disk for later use.

The package includes implementations of some of the most advanced likelihood-free inference techniques.

One example of these is BOLFI [2], which estimates the discrepancy function using Gaussian process regression and uses Bayesian optimization for parameter search, which has recently been shown to accelerate likelihood-free inference up to several orders of magnitude.

Main Features

Ease of use
The library has intuitive user-interface, allowing the researcher to get feasibility studies done quickly. Interfacing eg. C++ simulators is also rather simple.

Inference tasks defined with graphical models
Suitable for various different inference tasks that can be defined as a DAG.

Native support for parallelization
The inference engine is built on top of *Dask*, a Python library for parallelizing computation. This enables the researcher to easily port the model to eg. a cluster computer without changes to the code.

Implemented inference methods
The library has ready-made implementations of standard ABC inference methods, such as rejection sampling and SMC-ABC, as well as some novel methods, such as BOLFI.

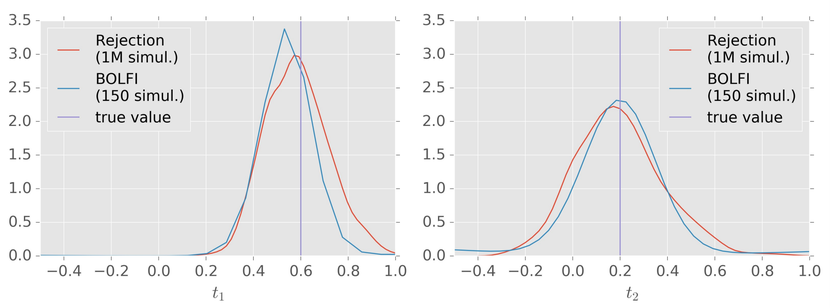
Storing inference results to disc
The library has support for storing e.g. simulated data and inference results to disc for later use.

Open source
The library is published under the BSD3 license, and credit will be given to authors contributing to the library. Authors can get visibility to their work by publishing them as a part of a larger open-source package.

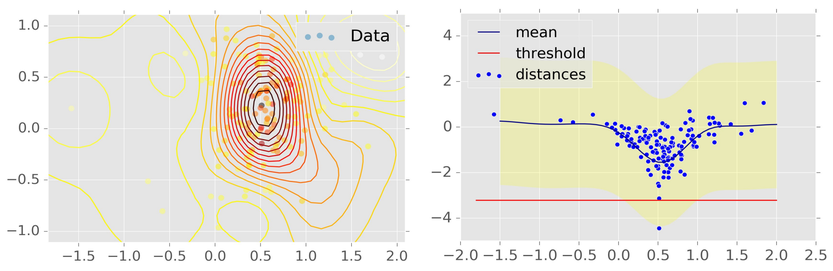
User Interface

Example of the user interface of the library:

```
import elfi
from elfi.ma2 import acov1, acov2, L2dist

# User-defined simulator
def MA2(t1, t2, n_sim=1, prng=None):
    n_obs = 100
    w = prng.randn(n_sim, n_obs+2)
    y = w[:,2:] + t1 * w[:,1:-1] + t2 * w[:,:-2]
    return y

# Observed data
y_obs = MA2(0.6, 0.2)

# Model definition
t1 = elfi.Prior('t1', 'uniform', -1, 2)
t2 = elfi.Prior('t2', 'uniform', -1, 2)
Y = elfi.Simulator('MA2', MA2,
                   t1, t2, observed=y_obs)
S1 = elfi.Summary('S1', acov1, Y)
S2 = elfi.Summary('S2', acov2, Y)
d = elfi.Discrepancy('d', L2dist, S1, S2)

# ABC posteriors
```




```
# BOLFI illustrations
```



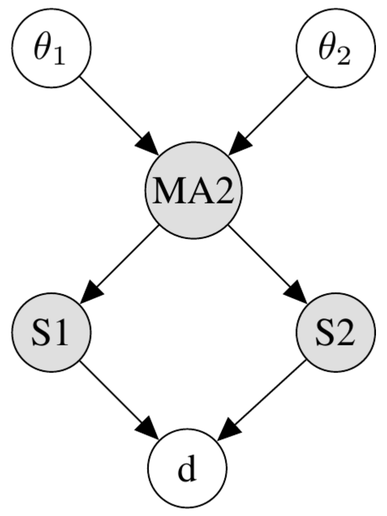

```
# Inference with Rejection ABC
N = 1000
rej = elfi.Rejection(d, [t1, t2], batch_size=N)
results = rej.sample(N, quantile=0.001)

# Change to log of distance
logL2 = lambda s1,s2: np.log(L2dist(s1,s2))
d = d.change_to(
    elfi.Discrepancy('log_d', logL2, S1, S2) )

# Inference with BOLFI [2]
gp_model = elfi.GPyModel(bounds=((-2,2), (-1,1))

bolfi = elfi.BOLFI(d, [t1, t2], batch_size=5,
                   n_surrogate_samples=150,
                   model=gp_model)

posterior = bolfi.infer()
```

Inference Tasks

ELFI allows the researcher to define the inference task as a graphical model. Nodes in the graphical model can be eg. Scipy distributions or user-defined functions.

$$x_t = w_t + \theta_1 w_{t-1} + \theta_2 w_{t-2}$$

User-defined generative model for simulated data (MA2)



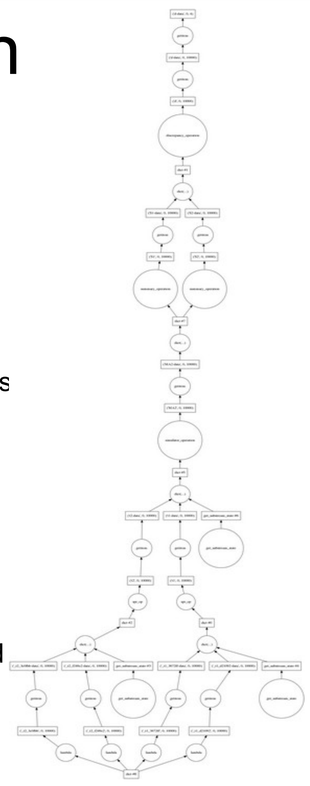
Example of a graphical model with two summary statistics (S1 and S2) and a discrepancy node (d)

Parallelization

The calculations required for performing the inference task are automatically transformed into a sequence of operations, part of which can be computed in parallel.

One such sequence of operations is visualized on the right. It visualizes the operations for computing distances from the example model.

We use the Dask library for scheduling the operations. The library works on desktop computers, but can be as well used in cluster environments, such as SLURM.



Take-home Messages

An open-source Python library for likelihood-free inference.

Easy to use, fast to do feasibility studies.

Multiple inference methods already implemented.

Can be interfaced with external simulators (e.g. C++).

Samples and inference results can be easily stored to disc.

Native parallelization, easy to up-scale inference tasks from desktop to cluster environments.

[1] Lintusaari et al. Fundamentals and Recent Developments in Approximate Bayesian Computation. Systematic Biology 2016.
[2] Gutmann & Corander. Bayesian Optimization for Likelihood-Free Inference of Simulator-Based Statistical Models. JMLR 2016.

Web page: http://elfi.readthedocs.io