ELFI Documentation

Release 0.5.0

ELFI authors

May 19, 2017

Contents

Contents:

CHAPTER 1

ELFI - Engine for Likelihood-Free Inference

ELFI is a statistical software package written in Python for Approximative Bayesian Computation (ABC), also known e.g. as likelihood-free inference, simulator-based inference, approximative Bayesian inference etc. This is useful, when the likelihood function is unknown or difficult to evaluate, but a generative simulator model exists.

The probabilistic inference model is defined as a directed acyclic graph, which allows for an intuitive means to describe inherent dependencies in the model. The inference pipeline is automatically parallelized from multiple cores up to a cluster environment. ELFI also handles seeding the random number generators and storing of the generated data for you so that you can easily repeat or fine tune your inference. Additionally, the package includes functionality for visualization.

Currently implemented ABC methods:

- rejection sampler
- Sequential Monte Carlo ABC sampler
- Bayesian Optimization for Likelihood-Free Inference (BOLFI) framework

Other notable included algorithms and methods:

- Bayesian Optimization
- No-U-Turn-Sampler, a Hamiltonian Monte Carlo MCMC sampler

GitHub page: https://github.com/elfi-dev/elfi

See examples under the notebooks directory to get started. Limited user-support may be asked from elfi-support.at.hiit.fi, but the Gitter chat is preferable.

Licenses:

- Code: BSD3
- Documentation: CC-BY 4.0

CHAPTER 2

Installation

To install ELFI, run this command in your terminal:

```
pip install elfi
```

If you don't have pip installed, this Python installation guide can guide you through the process.

ELFI is currently tested only with Python 3.5. If you are new to Python, perhaps the simplest way to install it is Anaconda

Optional dependencies

Optionally you may wish to install also the following packages:

- *graphviz* for drawing graphical models (Graphviz must be installed separately)

Virtual environment using Anaconda

If you want to create a virtual environment before installing, you can do so with Anaconda:

```
conda create -n elfi python=3.5 scipy
source activate elfi
pip install elfi
```

Potential problems with installation

ELFI depends on several other Python packages, which have their own dependencies. Resolving these may sometimes go wrong:

- If you receive an error about missing *numpy*, please install it first.

- If you receive an error about *yaml.load*, install *pyyaml*.
- On OS X with Anaconda virtual environment say *conda install python.app* and then use *pythonw* instead of *python*.
- Note that ELFI currently supports Python 3.5 only, although 3.x may work as well.

From sources

The sources for ELFI can be downloaded from the Github repo.

You can either clone the public repository:

```
git clone https://github.com/elfi-dev/elfi.git
```

Or download the tarball:

```
curl  -OL https://github.com/elfi-dev/elfi/tarball/master
```

Note that for development it is recommended to base your work on the *dev* branch instead of *master*.

Once you have a copy of the source, you can install it with:

```
pip install -e .
```

This will install ELFI along with its default requirements.

CHAPTER 3

Usage

To use ELFI in a project:

```
import elfi
```

For tutorials, please see the Jupyter Notebooks under the notebooks directory. Feel free to add your own in the zoo.

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/elfi-dev/elfi/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

ELFI could always use more documentation, whether as part of the official ELFI docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/elfi-dev/elfi/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *ELFI* for local development.

1. Fork the *elfi* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/elfi.git
```

3. Install your local copy and the development requirements into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development. Due to a bug in the pip installation of GPy numpy needs to be installed manually.:

```
$ mkvirtualenv elfi
$ cd elfi/
$ pip install numpy
$ make dev
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. Follow the *Style Guidelines*
6. When you're done making changes, check that your changes pass flake8 and the tests:

```
$ make lint
$ make test
```

   Also make sure that the docstrings of your code are formatted properly:

```
$ make docs
```

7. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

8. Submit a pull request through the GitHub website.

Style Guidelines

The projects follows the Khan Academy Style Guide. Except that we use numpy style docstrings instead of Google style docstrings.

See this example for how to format the docstrings.

Additional Style Guidelines

- Use the `.format()` string method instead of the old percent operator. For more information see PyFormat.
- Use the type hinting syntax suggested here in the docstrings.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.5 and later. Check https://travis-ci.org/elfi-dev/elfi/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ py.test tests.test_elfi
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search