mcmc Documentation

Release 0.1.0

Xinyang Li

October 27, 2016

Contents

Contents:

mcmc —- Markov Chain Monte Carlo

A small program for Markov chain Monte Carlo simulations.

1.1 Features

Currently you can run the program from anywhere you like by typing 'python3 path_to_source_code_folder/mcmc.py'. Output files will saved to the folder where you run the command. There are four parameters for this program,

k ———- the number of nodes

r ———- the weight coefficient

T ———- temperature

nsteps —– number of total steps (graphs).

You can modify those parameters in mcmc/mcmc.py

- Note: The simulations is highly sensitive to temperature. High temperature leads to fast convergence, while less stable distribution, i.e., you will get more kinds of graphs. On the other hand, low temperature causes several times slower convergence and failing in ergodicity assumption.
- Ergodicity can be verified with smaller number of nodes, e.g., 5 or 6, and a reasonable number of steps.

The program will produce several output files.

The list of all edges will be written into 'edgelist'.

The file 'output' contains number of steps, average degree of node 0, average number of edges, and average max length of all shortest paths from 0, respectively, and they are stored step by step to show the time evolution, which can also be used to check convergence.

The arbitrary graph needed to be drawn is named 'output.png', while most probable one is called 'top.png'. Note that I do not really count whether it is top 1% or not. Instead, I simply draw the one with highest histogram. In general, this range is tighter than 1%.

The sorted histograms are stored in 'sorted_histogram', where the columns are indices, edges and histograms, respectively.

Some summarized information is saved to 'summary' for your convenience.

- Free software: MIT license
- Documentation: latest

1.2 Todo

- Some improvements on draw.py.

1.3 Credits

This package was created with Cookiecutter and the audreyr/**cookiecutter-pypackage_** project template.

CHAPTER 2

Installation

2.1 Stable release

To install mcmc, run this command in your terminal:

```
$ pip install mcmc
```

This is the preferred method to install mcmc, as it will always install the most recent stable release.

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

2.2 From sources

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

You can either clone the public repository:

```
$ git clone git://github.com/tautomer/mcmc
```

Or download the tarball:

```
$ curl  -OL https://github.com/tautomer/mcmc/tarball/master
```

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

```
$ python setup.py install
```

Usage

To use mcmc in a project:

```
python3 path_to_source_code_folder/mcmc.py
```

To test the code, tyoe:

```
nosetests -v
```

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/tautomer/mcmc/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.

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/tautomer/mcmc/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 :)

4.2 Get Started!

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

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

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

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv mcmc
$ cd mcmc/
$ python setup.py develop
```

4. Create a branch for local development:

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

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 mcmc tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. 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
```

7. Submit a pull request through the GitHub website.

4.3 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 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/tautomer/mcmc/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_mcmc
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search