Longstaff-Schwartz Algorithm Documentation

Release 0.2.0

luphord

Nov 10, 2022

Contents:

CHAPTER 1

Longstaff-Schwartz Algorithm

A Python implementation of the Longstaff-Schwartz linear regression algorithm for the evaluation of call rights and American options.

- Seminal paper: **Francis A. Longstaff, Eduardo S. Schwartz**, *Valuing American Options by Simulation: A Simple Least-Squares Approach* (The Review of Financial Studies) (2001) Vol 14, No 1, pp. 113-147
- Documentation: https://longstaff-schwartz.readthedocs.io
- Free software: MIT license

1.1 Talks

- PyConDE 2019-10-10: Slides, Jupyter Notebook
- PyData Meetup 2019-09-18: Slides, Jupyter Notebook

1.2 Usage

```
from longstaff_schwartz.algorithm import longstaff_schwartz
from longstaff_schwartz.stochastic_process import GeometricBrownianMotion
import numpy as np

# Model parameters
t = np.linspace(0, 5, 100)  # timegrid for simulation
r = 0.01  # riskless rate
sigma = 0.15  # annual volatility of underlying
```

(continues on next page)

(continued from previous page)

```
n = 100  # number of simulated paths

# Simulate the underlying
gbm = GeometricBrownianMotion(mu=r, sigma=sigma)
rnd = np.random.RandomState(1234)
x = gbm.simulate(t, n, rnd)  # x.shape == (t.size, n)

# Payoff (exercise) function
strike = 0.95

def put_payoff(spot):
    return np.maximum(strike - spot, 0.0)

# Discount factor function
def constant_rate_df(t_from, t_to):
    return np.exp(-r * (t_to - t_from))

# Approximation of continuation value
def fit_quadratic(x, y):
    return np.polynomial.Polynomial.fit(x, y, 2, rcond=None)

# Selection of paths to consider for exercise
# (and continuation value approxmation)
def itm(payoff, spot):
    return payoff > 0

# Run valuation of American put option
npv_american = longstaff_schwartz(x, t, constant_rate_df,
                                  fit_quadratic, put_payoff, itm)

# European put option for comparison
npv_european = constant_rate_df(t[0], t[-1]) * put_payoff(x[-1]).mean()

# Check results
assert np.round(npv_american, 4) == 0.0734
assert np.round(npv_european, 4) == 0.0626
assert npv_american > npv_european
```

1.3 Plots

For details see PyData Meetup Jupyter Notebook.

1.3.1 Approximation of continuation value

1.3.2 Favourable exercise

1.4 Credits

Main developer is luphord.

Primary source for the algorithm is **Francis A. Longstaff, Eduardo S. Schwartz**, *Valuing American Options by Simulation: A Simple Least-Squares Approach* (The Review of Financial Studies) (2001) Vol 14, No 1, pp. 113-147. There is no affiliation between the authors of the paper and this code.

This package was prepared with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install Longstaff-Schwartz Algorithm, run this command in your terminal:

```
$ pip install longstaff_schwartz
```

This is the preferred method to install Longstaff-Schwartz Algorithm, 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 Longstaff-Schwartz Algorithm can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/luphord/longstaff_schwartz
```

Or download the tarball:

```
$ curl  -OL https://github.com/luphord/longstaff_schwartz/tarball/master
```

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

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

CHAPTER 3

Usage

To use Longstaff-Schwartz Algorithm in a project:

```
import longstaff_schwartz
```

CHAPTER 4

longstaff_schwartz

4.1 longstaff_schwartz package

4.1.1 Submodules

4.1.2 longstaff_schwartz.algorithm module

4.1.3 longstaff_schwartz.binomial module

4.1.4 longstaff_schwartz.cli module

4.1.5 longstaff_schwartz.regression_basis module

4.1.6 longstaff_schwartz.stochastic_process module

4.1.7 Module contents

A Python implementation of the Longstaff-Schwartz linear regression algorithm for the evaluation of call rights and American options.

CHAPTER 5

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:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/luphord/longstaff_schwartz/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.

5.1.2 Fix Bugs

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

5.1.3 Implement Features

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

5.1.4 Write Documentation

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

5.1.5 Submit Feedback

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

5.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/longstaff_schwartz.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 longstaff_schwartz
$ cd longstaff_schwartz/
$ 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 longstaff_schwartz 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.

5.3 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 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.com/github/luphord/longstaff_schwartz/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

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

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 6

Credits

6.1 Development Lead

- luphord <luphord@protonmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.2.0 (2022-11-10)

- Drop support for Python 3.7
- Add support for Python 3.9 and 3.10
- Upgrade dependencies
- Reformat code with black
- Migrate from travis-ci.com to GitHub actions
- Upgrade development status to alpha

7.2 0.1.1 (2020-12-01)

- Support Python 3.8
- Migrate to travis-ci.com
- Increase number of simulated paths in example to prevent poor conditioning warning

7.3 0.1.0 (2019-10-03)

- First release on PyPI.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

l

Index

L