deepiv Documentation

Release 0.1.0

Jason Hartford

Nov 20, 2017

Contents

Contents:

CHAPTER 1

DeepIV

IMPORTANT: Newer versions of Keras have broken this implementation. This code currently only support Keras 2.0.6 (which is what will be installed if you use the pip install instructions described below).

A package for counterfactual prediction using deep instrument variable methods that builds on Keras.

You can read how the method works in our DeepIV paper.

If you use this package in your research, please cite it as:

```
@inproceedings{Hartford17,
author    = {Jason Hartford and
        Greg Lewis and
        Kevin Leyton-Brown and
        Matt Taddy},
title     = {Deep IV: A Flexible Approach for Counterfactual Prediction},
booktitle = {Proceedings of the 34th International Conference on Machine Learning,
        {ICML} 2017, Sydney, Australia, 6-11 August 2017},
pages     = {1--9},
year      = {2017}
}
```

- Free software: MIT license
- Documentation: https://deepiv.readthedocs.io.

1.1 Installation

To use DeepIV, you can simply naviage to to the DeepIV directory on your machine and run:

pip install .

You can then use the package by simply running: import deepiv in python. See the examples directory for example usage.

The package is currently under active development, so you may want to install it using the following command:

> pip install -e .

By doing this, every time you git pull an update, it will be reflected in your installation.

1.2 Usage

The DeepIV package is simply a subclass of the Keras Model class that provides the necessary functions for fitting Deep instrumental variable models. Because of this, you can think of it as a drop-in replacement of the Keras Model object. The DeepIV procedure consists of two stages: 1. Fit the Treatment model. 2. Fit the Response model that takes the fitted Treatment model as input.

Example usage is shown in the experiments directory.

`demand_simulation.py` gives a simple example using a feedforward network for both the treatment and the response models.

`demand_simulation_mnist.py` is a little more complicated: it uses a convolutation network to fit an image embedding and then concatinates the embedding with other features to fit the network.

Both those examples use simulated data where ground truth is known, so they can report the causal mean squared error. On real data this isn't possible, so we advise that you use a holdout set to tune hyperparameters of the network (or cross validation in the case of small networks). You can choose hyperparameters based on the losses returned at each stage (see the paper for details on why this works).

DeepIV should be compatable with all Keras layers, so the Keras documentation is a good place to learn about designing network architectures. Feel free to file a bug report if something doesn't work.

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 deepiv, run this command in your terminal:

```
$ pip install deepiv
```

This is the preferred method to install deepiv, 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 deepiv can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/jhartford/deepiv
```

Or download the tarball:

```
$ curl  -OL https://github.com/jhartford/deepiv/tarball/master
```

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

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

CHAPTER 3

Usage

To use deepiv in a project:

```
import deepiv
```

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/jhartford/deepiv/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

deepiv could always use more documentation, whether as part of the official deepiv 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/jhartford/deepiv/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 *deepiv* for local development.

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

   ```
   $ git clone git@github.com:your_name_here/deepiv.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 deepiv
   $ cd deepiv/
   $ 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 deepiv 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/jhartford/deepiv/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_deepiv
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search