
pyBAMBI Documentation

Release 0.2.0

Dark Machines

Feb 05, 2020

Contents

1	Quick links	3
2	Notes	5
3	Installation instructions	7
4	pybambi package	9
5	pybambi	15
6	pyBAMBI	17
	Python Module Index	19
	Index	21

pyBAMBI Resurrecting BAMBI for the pythonic deep learning era

Author Dark Machines collaboration

Organiser Will Handley

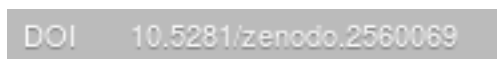
Version 0.2.0

GitHub <https://github.com/DarkMachines/pyBAMBI>

Documentation <https://pybambi.readthedocs.io/>

Website <https://darkmachines.org>

Paper <https://arxiv.org/abs/1110.2997>



CHAPTER 1

Quick links

- [Contributing](#)
- [Code of conduct](#)

CHAPTER 2

Notes

Currently, there is a test script in `test.py`, which shows how to call `pyBAMBI`. The main piece of work to be done is to implement neural network modelling in `pybambi/dumper.py` from the view onto the live points every `nlive` iterations.

Installation instructions

- [MultiNest](#) installation
- [PolyChord](#) installation

You can clone the repository with

```
git clone git@github.com:DarkMachines/pyBAMBI.git
```

Although if you wish to [contribute](#), you can either create a new branch and push (if you are a member of the [DarkMachines](#) sampling community) or fork the repository to your own account.

You can run the tests with either of the commands:

```
python -m pytest tests
python setup.py test
```


4.1 Subpackages

4.1.1 pybambi.neuralnetworks package

Submodules

pybambi.neuralnetworks.base module

Base predictor class.

Author: Will Handley (wh260@cam.ac.uk) Date: November 2018

class pybambi.neuralnetworks.base.**Predictor** (*params, logL, split=0.8*)

Bases: `object`

Base predictor class.

This takes in a training set params -> logL, and aims to construct a mapping between them.

Parameters

- **params** – *numpy.array* of physical parameters to train on shape (ntrain, ndims)
- **logL** – *numpy.array* of loglikelihoods to learn shape (ntrain,)

uncertainty ()

Uncertainty value for the trained model.

valid (*loglikelihood*)

Check validity of proxy.

Checks to see if the supplied log likelihood value is within the current range of likelihoods, including the uncertainty

Parameters **loglikelihood** – Value of the log likelihood that needs checking

pybambi.neuralnetworks.kerasnet module

Keras neural net predictor.

This implements a Keras Sequential model (a deep MLP)

Author: Martin White (martin.white@adelaide.edu.au) Date: December 2018

```
class pybambi.neuralnetworks.kerasnet.KerasNetInterpolation (params,          logL,  
                                                         split=0.8,  
                                                         model=None)
```

Bases: `pybambi.neuralnetworks.base.Predictor`

Keras neural net interpolation.

Returns the loglikelihood from a Keras neural net-based interpolator

Trains a basic 3-layer neural network with 200 neurons per layer.

Parameters

- **params** – *numpy.array* of physical parameters to train on shape (ntrain, ndims)
- **logL** – *numpy.array* of loglikelihoods to learn shape (ntrain,)

uncertainty ()

Uncertainty value for the trained keras model.

pybambi.neuralnetworks.nearestneighbour module

Nearest neighbour interpolation predictor.

Author: Will Handley (wh260@cam.ac.uk) Date: November 2018

This implements a nearest neighbour interpolation, and is designed as a placeholder predictor, rather than an actual neural network

```
class pybambi.neuralnetworks.nearestneighbour.NearestNeighbourInterpolation (params,  
                                                                              logL,  
                                                                              split=0.8)
```

Bases: `pybambi.neuralnetworks.base.Predictor`

Nearest Neighbour interpolation.

Returns the loglikelihood of the training point closest in parameter space

Parameters

- **params** – *numpy.array* of physical parameters to train on shape (ntrain, ndims)
- **logL** – *numpy.array* of loglikelihoods to learn shape (ntrain,)

uncertainty ()

Rough uncertainty for the nearest neighbour model.

Module contents

Collection of neural network interpolators.

Author: Will Handley (wh260@cam.ac.uk) Date: November 2018

4.2 Submodules

4.3 pybambi.bambi module

Driving routine for pyBAMBI.

Author: Will Handley (wh260@cam.ac.uk) Date: November 2018

`pybambi.bambi.run_pyBAMBI` (*loglikelihood*, *prior*, *nDims*, ***kwargs*)
Run pyBAMBI.

Parameters

- **nested_sampler** (*str*) – Choice of nested sampler. Options: [*'multinest'*, *'polychord'*]. Default *'polychord'*.
- **nlive** (*int*) – Number of live points. Default *nDims**25
- **root** (*str*) – root of filename. Default *'chains/<nested_sampler>'*
- **num_repeats** (*int*) – number of repeats for polychord. Default *nDims**5
- **eff** (*float*) – efficiency for multinest. Default *0.5**nDims*
- **learner** (*object*) – information indicating what learning algorithm to use for approximating the likelihood. Can be the string *'keras'*, or a *keras.models.Model* Default *'keras'*
- **ntrain** (*int*) – Number of training points to use Default *nlive*
- **proxy_tolerance** (*float*) – Required accuracy of proxy. Default *0.01*
- **ns_output** (*int*) – Nested sampling output level.

4.4 pybambi.manager module

BAMBI management object.

Author: Pat Scott (p.scott@imperial.ac.uk) Date: Feb 2019

class `pybambi.manager.BambiManager` (*loglikelihood*, *learner*, *proxy_tolerance*, *failure_tolerance*, *ntrain*)

Bases: `object`

Does all the talking for BAMBI.

Takes a new set of training data from the dumper and trains (or retrains) a neural net, and assesses whether or not it can be used for a given parameter combination.

Parameters **ntrain** (*int*) – Number of training points to use

dumper (*live_params*, *live_loglks*, *dead_params*, *dead_loglks*)
Respond to signal from nested sampler.

loglikelihood (*params*)
Bambi Proxy wrapper for original loglikelihood.

make_learner (*params*, *loglikes*)
Construct a Predictor.

train_new_learner (*params*, *loglikes*)
Train a new Predictor.

4.5 pybambi.multinest module

Wrapper for PyMultiNest.

Author: Will Handley (wh260@cam.ac.uk) Date: November 2018

```
pybambi.multinest.run_multinest(loglikelihood, prior, dumper, nDims, nlive, root, ndump, eff,  
                                seed=-1)
```

Run MultiNest.

See <https://arxiv.org/abs/0809.3437> for more detail

Parameters

- **loglikelihood** (*callable*) – probability function taking a single parameter:
 - **theta**: `numpy.array` physical parameters, *shape*=(*nDims*,)
returning a log-likelihood (float)
- **prior** (*callable*) – tranformation function taking a single parameter
 - **cube**: `numpy.array` hypercube parameters, *shape*=(*nDims*,)
returning physical parameters (*numpy.array*)
- **dumper** (*callable*) – access function called every *nlive* iterations giving a window onto current live points. Single parameter, no return:
 - **live**: *numpy.array* live parameters and loglikelihoods, *shape*=(*nlive*,*nDims*+1)
- **nDims** (*int*) – Dimensionality of sampling space
- **nlive** (*int*) – Number of live points
- **root** (*str*) – base name for output files
- **ndump** (*int*) – How many iterations between dumper function calls
- **eff** (*float*) – Efficiency of MultiNest
- **seed** (*int*) – Seed for sampler. Optional, no default seed.

4.6 pybambi.polychord module

Wrapper for PyPolyChord.

Author: Will Handley (wh260@cam.ac.uk) Date: November 2018

```
pybambi.polychord.run_polychord(loglikelihood, prior, dumper, nDims, nlive, root, ndump,  
                                num_repeats, seed=-1)
```

Run PolyChord.

See <https://arxiv.org/abs/1506.00171> for more detail

Parameters

- **loglikelihood** (*callable*) – probability function taking a single parameter:
 - **theta**: `numpy.array` physical parameters, *shape*=(*nDims*,)
returning a log-likelihood (float)
- **prior** (*callable*) – tranformation function taking a single parameter
 - **cube**: `numpy.array` hypercube parameters, *shape*=(*nDims*,)

returning physical parameters (*numpy.array*)

- **dumper** (*callable*) – access function called every *nlive* iterations giving a window onto current live points. Single parameter, no return:
 - **live**: *numpy.array* of live parameters and loglikelihoods, *shape*=(*nlive*,*nDims*+1)
- **nDims** (*int*) – Dimensionality of sampling space
- **nlive** (*int*) – Number of live points
- **root** (*str*) – base name for output files
- **ndump** (*int*) – How many iterations between dumper function calls
- **num_repeats** (*int*) – Length of chain to generate new live points
- **seed** (*int*) – Seed for sampler. Optional, no default seed.

4.7 Module contents

Main pyBAMBI module.

Author: Will Handley (wh260@cam.ac.uk) Date: November 2018

4.7.1 Functions

- `run_pyBAMBI`

CHAPTER 5

pybambi

pyBAMBI Resurrecting BAMBI for the pythonic deep learning era

Author Dark Machines collaboration

Organiser Will Handley

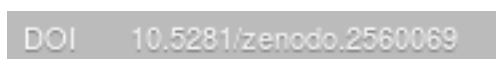
Version 0.2.0

GitHub <https://github.com/DarkMachines/pyBAMBI>

Documentation <https://pybambi.readthedocs.io/>

Website <https://darkmachines.org>

Paper <https://arxiv.org/abs/1110.2997>



6.1 Quick links

- [Contributing](#)
- [Code of conduct](#)

6.2 Notes

Currently, there is a test script in `test.py`, which shows how to call pyBAMBI. The main piece of work to be done is to implement neural network modelling in `pybambi/dumper.py` from the view onto the live points every `nlive` iterations.

6.3 Installation instructions

- [MultiNest installation](#)
- [PolyChord installation](#)

You can clone the repository with

```
git clone git@github.com:DarkMachines/pyBAMBI.git
```

Although if you wish to [contribute](#), you can either create a new branch and push (if you are a member of the [DarkMachines](#) sampling community) or fork the repository to your own account.

You can run the tests with either of the commands:

```
python -m pytest tests
python setup.py test
```

p

- `pybambi`, [13](#)
- `pybambi.bambi`, [11](#)
- `pybambi.manager`, [11](#)
- `pybambi.multinest`, [12](#)
- `pybambi.neuralnetworks`, [10](#)
- `pybambi.neuralnetworks.base`, [9](#)
- `pybambi.neuralnetworks.kerasnet`, [10](#)
- `pybambi.neuralnetworks.nearestneighbour`,
[10](#)
- `pybambi.polychord`, [12](#)

B

BambiManager (class in `pybambi.manager`), 11

D

dumper() (`pybambi.manager.BambiManager` method), 11

K

KerasNetInterpolation (class in `pybambi.neuralnetworks.kerasnet`), 10

L

loglikelihood() (`pybambi.manager.BambiManager` method), 11

M

make_learner() (`pybambi.manager.BambiManager` method), 11

N

NearestNeighbourInterpolation (class in `pybambi.neuralnetworks.nearestneighbour`), 10

P

Predictor (class in `pybambi.neuralnetworks.base`), 9

`pybambi` (module), 13

`pybambi.bambi` (module), 11

`pybambi.manager` (module), 11

`pybambi.multinest` (module), 12

`pybambi.neuralnetworks` (module), 10

`pybambi.neuralnetworks.base` (module), 9

`pybambi.neuralnetworks.kerasnet` (module), 10

`pybambi.neuralnetworks.nearestneighbour` (module), 10

`pybambi.polychord` (module), 12

R

run_multinest() (in module `pybambi.multinest`), 12

run_polychord() (in module `pybambi.polychord`), 12

run_pyBAMBI() (in module `pybambi.bambi`), 11

T

train_new_learner() (`pybambi.manager.BambiManager` method), 11

U

uncertainty() (`pybambi.neuralnetworks.base.Predictor` method), 9

uncertainty() (`pybambi.neuralnetworks.kerasnet.KerasNetInterpolation` method), 10

uncertainty() (`pybambi.neuralnetworks.nearestneighbour.NearestNeighbourInterpolation` method), 10

V

valid() (`pybambi.neuralnetworks.base.Predictor` method), 9