PHETS Documentation

Release 1.0

Dec 07, 2017

CONTENTS

This package offers high-level tools for exploration and visualization of delay coordinate embedding and persistent homology. It is used to investigate the utilization of these tools together as a signal processing technique.

PHETS encompasses four submodules:

- *signals*
- phomology
- embed
- prfstats

signals holds the TimeSeries and Trajectory classes, which can be initialized from arrays or text files. Calling the embed method of a TimeSeries returns a Trajectory; calling the project method of Trajectory returns a TimeSeries. TimeSeries and Trajectory both inherit from BaseTrajectory, where all cropping, windowing, and normalization is handled.

phomology holds the Filtration class, which is initialized from a Trajectory and a dict of filtration parameters. Filtration movies, persistence diagrams, and persistence rank functions are created by calling the respective methods of the Filtration class.

embed holds the embed function, as well as functions for generating movies. The movies functions take one or more TimeSeries and return one or more Trajectory objects (created in the process of building the movies).

prfstats holds functions for statistical analysis of PRFs. Generally, they take one or two Trajectory objects, create PRFs from the windows of the the Trajectory objects, do some analysis, and then save plots from the results.

CHAPTER

ONE

TROUBLESHOOTING

1.1 matplotlib backend error

Comment out the first line of `PHETS/matplotlibrc`

1.2 compiling `find_landmarks.c` on OSX

PHETS requires the OpenMP C library `omp.h`. From what I can tell, OpenMP is not included in clang (the default C compiler on macOS), and may only be installed /configured for recent versions, and not with great ease. For these reasons, we've never tried to run PHETS on clang, and cannot guarantee it will work correctly.

On the other hand, OpenMP works with gcc out of the box, and you may already have a version of gcc installed. If so, determine the version and edit `find_landmarks_c_compile_str` in `config.py` to match. (NOTE: on macOS, `gcc` is a symlink for clang. This is avoided by including the version number, eg `gcc-5`.)

If you do not have gcc installed, you can do `brew install gcc` and then, as above, tweak `config.py`. You can also tell brew to install a particular version if you would like (anything 5+ should work).

A quick way to test if things are working is to run `python refresh.py` in the PHETS directory. This script will remove a number of temporary files and attempt to compile `find_landmarks.c`

If the compiler is still giving errors (don't mind warnings), try `brew upgrade gcc` or `brew reinstall gcc --without-multilib`

See here and here for more information.

CHAPTER

TWO

REGRESSION TESTS

pytest is used for testing. To run the test suite, type `pytest --tb=short` from the top-level directory. (Running pytest within a subdirectory will only execute the tests for that submodule.)

Each submodule contains a `unit_test` directory. The tests themselves are defined in `unit_tests/test__<submodule>.py`. These are not exactly unit tests – rather, each one calls or initializes a user-facing feature and compares the result to a saved reference. The input data is found in `unit_tests/data` and the references in `<unit_tests/ref>`.

In the case of the `prfstats``module, in order to keep test execution time lwo, the input data is pre-computed sets of Filtration objects to keep test execution time low. A small, correct change to PHETS can break Python's ability to load these objects from file, breaking the tests. I this case, run the routines in ``prfstats/unit_tests/prepare__data.py`, and the tests should work correctly. Routines in `unit_tests/prepare__refs.py` should be run *only when you wish to change the behavior of existing functionality*. They should also be run and individually (that is, don't change the refs for features that you aren't intentionally modifying).

CHAPTER

THREE

THIS DOCUMENTATION

This documentation is built with Sphinx. The autodoc extension is used to generate the library reference from doc-strings in the Python code. The text and layout for all other sections (eg this paragraph) is defined in `docs/source/index.rst`.

To build this documentation, TeX must be installed, along with the following:

- texlive-latex-recommended
- texlive-fonts-recommended
- texlive-latex-extra
- latexmk

I used `sudo apt-get install <package>` for each.

Now, simply .. code-block:

```
cd docs
make latexpdf
```

The updated documentation is saved to `docs/latex/PHETS.pdf`.

CHAPTER

FOUR

REFERENCE

4.1 signals

class signals.**BaseTrajectory**(*data*, *crop=(None*, *None)*, *num_windows=None*, *window_length=None*, *vol_norm=(False*, *False*, *False)*, *time_units='samples'*, *name=None*, *fname=None*)

Parameters

- **data** (*str or array*) – The filename to load, or array. If a filename, sets `fname`.
- **crop** (*array, optional*) – Range of signal to work with. Observes `time_units`. Either or both bounds may be None. format: (start, stop). default: (None, None)
- **num_windows** (*int, optional*) – Slice signal into `windows` evenly spaced windows. default: None
- **window_length** (*int or float, optional*) – Observes `time_units` if None, `window_length == len(data) / num_windows` default: None
- **vol_norm** (*arr, optional*) – Normalize amplitude by (full, crop, window). default: (False, False, False)
- **time_units** (*str, optional*) – `'samples'` or `'seconds'` Observes `config.SAMPLE_RATE` default: `'samples'`
- **name** (*string, optional*) – Sets `name`, a label used for titles for plots. If None and `fname` is not None, `name` is derived from `fname`. default: None
- **fname** (*string, optional*) – If `data` is not a filename (i.e. is an array), sets `fname`. default: None

crop(*crop_cmd*)

Set `data` to the region of `data_full` specified by `crop_cmd` and `time_units`.

Parameters **crop_cmd** (*array*) – observes `time_units` format: (start, stop)

slice(*num_windows*, *window_length=None*)

Sets `windows`, an array of evenly spaced windows from `data`.

Parameters

- **num_windows** (*int*) –
- **window_length** (*int or float, optional*) – observes 'time_units'' if None, `window_length == len(data) / num_windows` default: None

class signals.**TimeSeries**(*data*, ***kwargs*)

Bases: `signals.signals.BaseTrajectory`

See *BaseTrajectory* for parameter descriptions

embed(*tau*, *m*)
Embed `data_full`, re-apply crop and slicing.

Parameters

- **tau** (*int or float*) – observes `time_units`
- **m** (*int*) –

Returns

Return type *Trajectory*

plot(*filename*)
Plot full time series with crop and windows demarcated, save to `filename`.

Parameters **filename** (*str*) –

plot_crop(*filename*)
Plot time series (crop only), save to `filename`. :param filename: :type filename: str

class signals.**Trajectory**(*data*, ***kwargs*)
Bases: `signals.signals.BaseTrajectory`

See *BaseTrajectory* for parameter descriptions

filtrations(*filt_params*, *quiet=True*, *status_str=None*)
Compute filtration for each window of trajectory.

Parameters

- **filt_params** (*dict*) – see Filtration
- **quiet** (*bool*) – terminal output noise

Returns array of Filtration objects

Return type array

project(*axis=0*)
Project `self.data_full` to time series, re-apply crop and slicing.

Parameters **axis** (*int*) –

Returns

Return type *TimeSeries*

4.2 phomology

class phomology.**Filtration**(*traj*, *params*, *silent=False*, *save=True*)

Parameters

- **traj** (`Trajectory`) – trajectory from which to build the filtration
- **params** (*dict*) – options for landmark selection, witness complex, distance modification, etc. see *build_filtration.build_filtration()*
- **silent** (*bool, optional*) – suppress stdout

- **save** (*bool or str, optional*) – Save the filtration to file for later use. If `save` is a path/filename, save filtration to `save`; `save` should end with `.p`. Otherwise, if `save` is True, save filtration to `phomology/filtrations/filt.p`. In this case, filtration may be loaded by calling *load_filtration()* without providing the filename parameter.

  default: True

intervals()

Returns birth and death times for holes in the complex filtration

Return type Intervals

movie(*filename*, ***kwargs*)
build filtration visualization

Parameters

- **filename** (*str*) – Output path/filename. Should end in '.mp4' or other movie format.
- **color_scheme** (*str, optional*) – `None`, `'highlight new'`, or `('birth time gradient', cycles)` where `cycles` is an `int` default: `None`
- **camera_angle** (*array*) – For 3D mode. (azimuthal, elevation) in degrees.

  default: (70, 45)
- **alpha** (*float*) – Opacity of simplexes

  default: 1
- **dpi** (*int*) – plot resolution – dots per inch

Returns

Return type None

pd()

Returns persistence diagram

Return type PD

plot_complex(*i*, *filename*, ***kwargs*)
plot complex at ith step of the filtration

Parameters

- **i** (*int*) –
- **filename** (*str*) – Output path/filename. Should end in '.png' or other supported image format.

Returns

Return type None

plot_pd(*filename*)
plot the persistence diagram

Parameters **filename** (*str*) – Output path/filename. Should end in '.png' or other supported image format.

Returns

Return type None

plot_prf(*filename*)
plot the persistence rank function

Parameters **filename** (*str*) – Output path/filename. Should end in '.png' or other supported image format.

Returns

Return type None

prf()

Returns persistence rank function

Return type PRF

phomology.**load_filtration**(*filename=None*)
load a filtration from file

Parameters **filename** (*str*) – Path/filename. Should end with .p

phomology.build_filtration.**build_filtration**(*input_file_name*, *parameter_set*, *silent=False*)

Parameters

- **input_file_name** (*str*) –
- **parameter_set** (*dict*) – Options for filtration and landmark selection. Defaults are set in config.py

  GENERAL:

  num_divisions Number of (epsilon) steps in filtration. The filtration parameter will be divided up equally in the interval [min_filtration_param, max_filtration_param].

  default: 50

  max_filtration_param The maximum value for the filtration parameter. If it is a negative integer, -x, the program will automatically choose the max filtration parameter such that the highest dimensional simplex constructed is of dimension x - 1.

  default: -20

  min_filtration_param The minimum value for the filtration parameter. Zero is usually fine.

  default: 0

  start How many lines to skip in the input file before reading data in.

  default: 0

  worm_length How many witnesses the program will read from the data file. If set to None, the program will read the file to the end. In general, a reasonable cap we have found is 10,000 witnesses and 200 or less landmarks.

  default: None

  ds_rate The ratio of number of witnesses / number of landmarks.

  default: 50

  landmark_selector "maxmin" How the landmarks are selected from among the witnesses. Only options are "EST" for equally spaced in time and "maxmin" for a maxmin distance algorithm.

  default: "maxmin"

WITNESS RELATION:

absolute The standard fuzzy witness relation says that a witness witnesses a simplex if the distance from the witness to each of the landmarks is within epsilon *more* than the distance to the closest landmark. If using the absolute relation, the closest landmark is dropped from the calculation, and the distance from a witness to each of the landmarks must be within epsilon of zero.

default: False

use_cliques If this is set to True, than witnesses are only used to connect edges, and higher simplices (faces, solids, etc.) are inferred from the 1-skeleton graph using the Bron-Kerbosch maximal clique finding algorithm. This can be useful in reducing noise if several of the false holes are triangles.

default: False

simplex_cutoff If not equal to zero, this caps the number of landmarks a witness can witness. Note: this does not effect automatic max_filtration_param selection.

default: 0

weak Uses a completely different relation. The filtration parameter k specifies that each witness will witness a simplex of its k-nearest neighbors. If this relation is used, max_filtration_param should be a positive integer, and num_divisions and min_filtration_param will be ignored.

default: False

use_twr Uses a completely different algorithm. TODO: insert your description here. Note: this works best with EST landmark selection. If max-min is used, be sure to set time_order_landmarks to True.

default: False

DISTANCE DISTORTIONS:

d_speed_amplify The factor by which distances are divided if the witness is at a relatively high speed.

default: 1

d_orientation_amplify The factor by which distances are divided if the witness and the landmark are travelling in similar directions.

default: 1

d_stretch The factor by which distances are divided if the vector from the witness to the landmark is in a similar direction (possibly backwards) as the direction in which time is flowing at the witness.

default: 1

d_ray_distance_amplify TODO: change this parameter. Right now, as long as the number is not 1, this will multiply the distance between two points by the distance between the closest points on the parameterized rays.

default: 1

d_use_hamiltonian If this is not zero, this will override all the above distortions. Distance will be computed using not only position coordinates, but also velocity coordinates. Velocity componnents are scaled by the value of this parameter (before squaring). If the value is negative, than the absolute value of the parameter is used, but the unit velocities are used instead of the actual velocities.

default: 0

use_ne_for_maxmin Whether or not to apply the above distance distortions to the max-min landmark selection (not recommended). Has no effect if landmark selector is EST.

default: False

MISC:

connect_time_1_skeleton If this is set to True, then on the first step of the filtration, each landmark will be adjoined by an edge to the next landmark in time. Note: this works best with EST landmark selection. If max-min is used, be sure to set time_order_landmarks to True.

default: False

reentry_filter Attempts to limit high dimensional simplices by requiring that landmarks get far away then come back. This only works if using cliques. Note: this works best with EST landmark selection. If max-min is used, be sure to set time_order_landmarks to True.

default: False

dimension_cutoff Simplexes with dimension greater than the dimension cuttoff will be seperated into their lower dimensional subsets when writing to the output file. This is very handy, as both Perseus and PHAT seem to take exponential time as a function of the dimension of a simplex. The caveat is that all homology greater than or equal to the dimension cutoff will be inacurate. Thus, if one cares about Betti 2, dimension cutoff should be at least 3.

still valid / in use ??? setting to 0 doesn't affect tests

default: 2

store_top_simplices If there is a dimension cutoff in use, this parameter determines at which point in the process the simplices are decomposed. By setting this to False, smaller simplices will be stored when they are discovered. This makes the output file a bit smaller, but takes a bit longer. The results will be left unchanged.

default: True

- **silent** (*bool*) – Suppress stdout

Returns (simplexes, (landmarks, witnesses), eps)

Return type array

4.3 embed

embed.**embed**(*data*, *tau*, *m*)

Parameters

- **data** (*array*) – one dimensional (time series)
- **tau** (*int*) – delay (samples)
- **m** (*int*) – target dimension

Returns m-dimensional embedding

Return type array

embed.movies.**compare_multi**()

embed.movies.**compare_vary_tau**()

embed.movies.**slide_window**(*ts*, *out_fname*, *m*, *tau*, *framerate=1*)

Build and save movie depicting the embedding of each window of *ts*.

Parameters

- **ts** (TimeSeries) – pre-windowed (e.g., initalized with *num_windows* or call *slice* method before passing to this function).
- **out_fname** (*str*) – filename
- **m** (*int*) – embedding dimension
- **tau** (*int or float*) – embedding delay, observes *ts.time_units*
- **framerate** (*int*) –

Returns

Return type *Trajectory*

embed.movies.**vary_tau**()

4.4 prfstats

PYTHON MODULE INDEX

INDEX