pythia Documentation

Release 0.3.0

Matthew Spellings

Feb 26, 2020

Contents

Pythia is a library to generate numerical descriptions of particle systems. Most methods rely heavily on freud for efficient neighbor search and other accelerated calculations.

CHAPTER 1

Installation

Pythia is available on PyPI as *pythia-learn*:

```
$ pip install pythia-learn freud-analysis
```

You can install pythia from source like this:

```
$ git clone https://github.com/glotzerlab/pythia.git
$ # now install
$ cd pythia && python setup.py install --user
```

Note: If using conda or a virtualenv, the *–user* argument in the pip command above is unnecessary.

CHAPTER 2

Citation

In addition to the citations referenced in the docstring of each function, we encourage users to cite the pythia project itself.

CHAPTER 3

Documentation

The documentation is available as standard sphinx documentation:

```
$ cd doc
$ make html
```

Automatically-built documentation is available at https://pythia-learn.readthedocs.io .

CHAPTER 4

Usage

In general, data types follow the hoomd-blue schema:

- Positions are an Nx3 array of particle coordinates, with *(0, 0, 0)* being the center of the box
- Boxes are specified as an object with *Lx*, *Ly*, *Lz*, *xy*, *xz*, and *yz* elements
- Orientations are specified as orientation quaternions: an Nx4 array of *(r, i, j, k)* elements

CHAPTER 5

Examples

Example notebooks are available in the *examples* directory:

- Unsupervised learning
- Supervised learning
- Steinhardt and Pythia order parameter comparison (FCC and HCP)

CHAPTER 6

Contents:

6.1 Bond-related Descriptors

This module computes relatively simple descriptors based on nearest-neighbor bonds, with few additional transformations.

pythia.bonds.**neighborhood_angle_singvals**(*box*, *positions*, *neighbors*, *rmax_guess=2.0*)

Construct a matrix of pairwise angles between *(rk - ri)* and *(rj - ri)* for all neighbors j and k(==j) of each particle i, for a particular number of neighbors. Returns the singular values of this matrix to fix permutation invariance.

This function uses the following citations:

```
@misc{freud2016,
        title = {freud},
        url = {https://doi.org/10.5281/zenodo.166564},
        abstract = {First official open-source release, includes a zenodo DOI for␣
↪citations.},
        author = {Harper, Eric S and Spellings, Matthew and Anderson, Joshua A␣
↪and Glotzer, Sharon C},
        month = nov,
        year = {2016},
        doi = {10.5281/zenodo.166564},
}
```

pythia.bonds.**neighborhood_angle_sorted**(*box*, *positions*, *neighbors*, *rmax_guess=2.0*)

Construct a matrix of pairwise angles between *(rk - ri)* and *(rj - ri)* for all neighbors j and k(==j) of each particle i, for a particular number of neighbors. Returns the sorted values of this matrix to fix permutation invariance.

This function uses the following citations:

```
@misc{freud2016,
        title = {freud},
        url = {https://doi.org/10.5281/zenodo.166564},
        abstract = {First official open-source release, includes a zenodo DOI for␣
↪citations.},
```

(continues on next page)

(continued from previous page)

```
        author = {Harper, Eric S and Spellings, Matthew and Anderson, Joshua A␣
↪and Glotzer, Sharon C},
        month = nov,
        year = {2016},
        doi = {10.5281/zenodo.166564},
}
```

pythia.bonds.**neighborhood_distance_singvals**(*box*, *positions*, *neighbors*, *rmax_guess=2.0*)

Construct a matrix of pairwise distances filled with |*r_k* - *r_j*| for all neighbors j and k(==j) of each particle i. Returns the singular values of this matrix to fix permutation invariance.

This function uses the following citations:

```
@misc{freud2016,
        title = {freud},
        url = {https://doi.org/10.5281/zenodo.166564},
        abstract = {First official open-source release, includes a zenodo DOI for␣
↪citations.},
        author = {Harper, Eric S and Spellings, Matthew and Anderson, Joshua A␣
↪and Glotzer, Sharon C},
        month = nov,
        year = {2016},
        doi = {10.5281/zenodo.166564},
}
```

pythia.bonds.**neighborhood_distance_sorted**(*box*, *positions*, *neighbors*, *rmax_guess=2.0*)

Construct a matrix of pairwise distances filled with |*r_k* - *r_j*| for all neighbors j and k(==j) of each particle i. Returns the sorted contents of this matrix to fix permutation invariance.

This function uses the following citations:

```
@misc{freud2016,
        title = {freud},
        url = {https://doi.org/10.5281/zenodo.166564},
        abstract = {First official open-source release, includes a zenodo DOI for␣
↪citations.},
        author = {Harper, Eric S and Spellings, Matthew and Anderson, Joshua A␣
↪and Glotzer, Sharon C},
        month = nov,
        year = {2016},
        doi = {10.5281/zenodo.166564},
}
```

pythia.bonds.**neighborhood_range_angle_singvals**(*box*, *positions*, *neigh_min*, *neigh_max*, *rmax_guess=2.0*)

Construct a matrix of pairwise angles between *(rk - ri)* and *(rj - ri)* for all neighbors j and k(==j) of each particle i, for a range of neighborhood sizes from neigh_min to neigh_max (inclusive). Returns the singular values of this matrix to fix permutation invariance.

This function uses the following citations:

```
@misc{freud2016,
        title = {freud},
        url = {https://doi.org/10.5281/zenodo.166564},
        abstract = {First official open-source release, includes a zenodo DOI for␣
↪citations.},
        author = {Harper, Eric S and Spellings, Matthew and Anderson, Joshua A␣
↪and Glotzer, Sharon C},
```

(continues on next page)

(continued from previous page)

```
        month = nov,
        year = {2016},
        doi = {10.5281/zenodo.166564},
}
```

pythia.bonds.**neighborhood_range_distance_singvals**(*box*, *positions*, *neigh_min*, *neigh_max*, *rmax_guess=2.0*)

Construct a matrix of pairwise distances filled with |*r_k* - *r_j*| for all neighbors j and k(==j) of each particle i, for a range of neighborhood sizes from neigh_min to neigh_max (inclusive). Returns the singular values of this matrix to fix permutation invariance.

This function uses the following citations:

```
@misc{freud2016,
        title = {freud},
        url = {https://doi.org/10.5281/zenodo.166564},
        abstract = {First official open-source release, includes a zenodo DOI for␣
↪citations.},
        author = {Harper, Eric S and Spellings, Matthew and Anderson, Joshua A␣
↪and Glotzer, Sharon C},
        month = nov,
        year = {2016},
        doi = {10.5281/zenodo.166564},
}
```

pythia.bonds.**normalized_radial_distance**(*box*, *positions*, *neighbors*, *rmax_guess=2.0*)

Returns the ratio of the euclidean distance of each near-neighbor to that of the nearest neighbor for each particle.

This function uses the following citations:

```
@misc{freud2016,
        title = {freud},
        url = {https://doi.org/10.5281/zenodo.166564},
        abstract = {First official open-source release, includes a zenodo DOI for␣
↪citations.},
        author = {Harper, Eric S and Spellings, Matthew and Anderson, Joshua A␣
↪and Glotzer, Sharon C},
        month = nov,
        year = {2016},
        doi = {10.5281/zenodo.166564},
}
```

6.2 Spherical Harmonic Descriptors

This module computes descriptors based on combinations of spherical harmonics applied to nearest-neighbor bonds.

pythia.spherical_harmonics.**abs_neighbor_average**(*box*, *positions*, *neigh_min=4*, *neigh_max=4*, *lmax=4*, *negative_m=True*, *reference_frame='neighborhood'*, *orientations=None*, *rmax_guess=1.0*, *noise_samples=0*, *noise_magnitude=0*, *nlist=None*)

Compute the neighbor-averaged spherical harmonics over the nearest-neighbor bonds of a set of particles. Returns the absolute value of the (complex) spherical harmonics

Parameters

- **neigh_min** – Minimum number of neighbor environment sizes to consider
- **neigh_max** – Maximum number of neighbor environment sizes to consider (inclusive)
- **lmax** – Maximum spherical harmonic degree l
- **negative_m** – Include negative m spherical harmonics in the output array?
- **reference_frame** – 'neighborhood': use diagonal inertia tensor reference frame; 'particle_local': use the given orientations array; 'global': do not rotate
- **orientations** – Per-particle orientations, only used when reference_frame == 'particle_local'
- **rmax_guess** – Initial guess of the distance to find *neigh_max* nearest neighbors. Only affects algorithm speed.
- **noise_samples** – Number of random noisy samples of positions to average the result over (disabled if 0)
- **noise_magnitude** – Magnitude of (normally-distributed) noise to apply to noise_samples different positions (disabled if *noise_samples == 0*)
- **nlist** – Freud neighbor list object to use (*None* to compute for neighbors up to *neigh_max*)

This function uses the following citations:

```
@misc{freud2016,
        title = {freud},
        url = {https://doi.org/10.5281/zenodo.166564},
        abstract = {First official open-source release, includes a zenodo DOI for␣
↪citations.},
        author = {Harper, Eric S and Spellings, Matthew and Anderson, Joshua A␣
↪and Glotzer, Sharon C},
        month = nov,
        year = {2016},
        doi = {10.5281/zenodo.166564},
}

@article{spellings2018,
        title = {Machine learning for crystal identification and discovery},
        volume = {64},
        url = {https://dx.doi.org/10.1002/aic.16157},
        doi = {10.1002/aic.16157},
        number = {6},
        journal = {AIChE Journal},
        author = {Spellings, Matthew and Glotzer, Sharon C},
        year = {2018},
        pages = {2198--2206},
}
```

pythia.spherical_harmonics.**abs_system_average**(*box*, *positions*, *neigh_min=4*, *neigh_max=4*, *lmax=4*, *negative_m=True*, *reference_frame='neighborhood'*, *orientations=None*, *rmax_guess=1.0*, *noise_samples=0*, *noise_magnitude=0*, *nlist=None*)

Compute the global-averaged spherical harmonics over the nearest-neighbor bonds of a set of particles. Returns the absolute value of the (complex) spherical harmonics

Parameters

- **neigh_min** – Minimum number of neighbor environment sizes to consider
- **neigh_max** – Maximum number of neighbor environment sizes to consider (inclusive)
- **lmax** – Maximum spherical harmonic degree l
- **negative_m** – Include negative m spherical harmonics in the output array?
- **reference_frame** – 'neighborhood': use diagonal inertia tensor reference frame; 'particle_local': use the given orientations array; 'global': do not rotate
- **orientations** – Per-particle orientations, only used when reference_frame == 'particle_local'
- **rmax_guess** – Initial guess of the distance to find *neigh_max* nearest neighbors. Only affects algorithm speed.
- **noise_samples** – Number of random noisy samples of positions to average the result over (disabled if 0)
- **noise_magnitude** – Magnitude of (normally-distributed) noise to apply to noise_samples different positions (disabled if *noise_samples == 0*)
- **nlist** – Freud neighbor list object to use (*None* to compute for neighbors up to *neigh_max*)

This function uses the following citations:

```
@misc{freud2016,
        title = {freud},
        url = {https://doi.org/10.5281/zenodo.166564},
        abstract = {First official open-source release, includes a zenodo DOI for␣
↪citations.},
        author = {Harper, Eric S and Spellings, Matthew and Anderson, Joshua A␣
↪and Glotzer, Sharon C},
        month = nov,
        year = {2016},
        doi = {10.5281/zenodo.166564},
}

@article{spellings2018,
        title = {Machine learning for crystal identification and discovery},
        volume = {64},
        url = {https://dx.doi.org/10.1002/aic.16157},
        doi = {10.1002/aic.16157},
        number = {6},
        journal = {AIChE Journal},
        author = {Spellings, Matthew and Glotzer, Sharon C},
        year = {2018},
        pages = {2198--2206},
}
```

pythia.spherical_harmonics.**bispectrum**(*box*, *positions*, *neighbors*, *lmax*, *rmax_guess=2.0*)

Computes bispectrum invariants of particle local environments. These are rotationally-invariant descriptions similar to a power spectrum of the spherical harmonics (i.e. steinhardt order parameters), but retaining more information.

Parameters

- **neighbors** – number of nearest-neighbors to consider for local environments

- **lmax** – maximum spherical harmonic degree to consider. O(lmax**3) descriptors will be generated.

This function uses the following citations:

```
@article{kondor2007,
        title = {A novel set of rotationally and translationally invariant␣
↪features for images based on the non-commutative bispectrum},
        url = {http://arxiv.org/abs/cs/0701127},
        journal = {arXiv:cs/0701127},
        author = {Kondor, Risi},
        month = jan,
        year = {2007},
}

@misc{freud2016,
        title = {freud},
        url = {https://doi.org/10.5281/zenodo.166564},
        abstract = {First official open-source release, includes a zenodo DOI for␣
↪citations.},
        author = {Harper, Eric S and Spellings, Matthew and Anderson, Joshua A␣
↪and Glotzer, Sharon C},
        month = nov,
        year = {2016},
        doi = {10.5281/zenodo.166564},
}
```

pythia.spherical_harmonics.**neighbor_average**(*box*, *positions*, *neigh_min=4*, *neigh_max=4*, *lmax=4*, *negative_m=True*, *reference_frame='neighborhood'*, *orientations=None*, *rmax_guess=1.0*, *noise_samples=0*, *noise_magnitude=0*, *nlist=None*)

Compute the neighbor-averaged spherical harmonics over the nearest-neighbor bonds of a set of particles. Returns the raw (complex) spherical harmonic values.

Parameters

- **neigh_min** – Minimum number of neighbor environment sizes to consider
- **neigh_max** – Maximum number of neighbor environment sizes to consider (inclusive)
- **lmax** – Maximum spherical harmonic degree l
- **negative_m** – Include negative m spherical harmonics in the output array?
- **reference_frame** – 'neighborhood': use diagonal inertia tensor reference frame; 'particle_local': use the given orientations array; 'global': do not rotate
- **orientations** – Per-particle orientations, only used when reference_frame == 'particle_local'
- **rmax_guess** – Initial guess of the distance to find *neigh_max* nearest neighbors. Only affects algorithm speed.
- **noise_samples** – Number of random noisy samples of positions to average the result over (disabled if 0)
- **noise_magnitude** – Magnitude of (normally-distributed) noise to apply to noise_samples different positions (disabled if *noise_samples == 0*)
- **nlist** – Freud neighbor list object to use (*None* to compute for neighbors up to *neigh_max*)

This function uses the following citations:

```
@misc{freud2016,
        title = {freud},
        url = {https://doi.org/10.5281/zenodo.166564},
        abstract = {First official open-source release, includes a zenodo DOI for␣
↪citations.},
        author = {Harper, Eric S and Spellings, Matthew and Anderson, Joshua A␣
↪and Glotzer, Sharon C},
        month = nov,
        year = {2016},
        doi = {10.5281/zenodo.166564},
}

@article{spellings2018,
        title = {Machine learning for crystal identification and discovery},
        volume = {64},
        url = {https://dx.doi.org/10.1002/aic.16157},
        doi = {10.1002/aic.16157},
        number = {6},
        journal = {AIChE Journal},
        author = {Spellings, Matthew and Glotzer, Sharon C},
        year = {2018},
        pages = {2198--2206},
}
```

pythia.spherical_harmonics.**steinhardt_q**(*box*, *positions*, *neighbors=12*, *lmax=6*, *rmax_guess=2.0*)

Compute a vector of per-particle Steinhardt order parameters.

Parameters

- **neighbors** – Number of neighbors (int) or maximum distance to find neighbors within (float)
- **lmax** – Maximum spherical harmonic degree l
- **rmax_guess** – Initial guess of the distance to find nearest neighbors, if appropriate. Only affects algorithm speed.

This function uses the following citations:

```
@misc{freud2016,
        title = {freud},
        url = {https://doi.org/10.5281/zenodo.166564},
        abstract = {First official open-source release, includes a zenodo DOI for␣
↪citations.},
        author = {Harper, Eric S and Spellings, Matthew and Anderson, Joshua A␣
↪and Glotzer, Sharon C},
        month = nov,
        year = {2016},
        doi = {10.5281/zenodo.166564},
}
```

pythia.spherical_harmonics.**system_average**(*box*, *positions*, *neigh_min=4*, *neigh_max=4*, *lmax=4*, *negative_m=True*, *reference_frame='neighborhood'*, *orientations=None*, *rmax_guess=1.0*, *noise_samples=0*, *noise_magnitude=0*, *nlist=None*)

Compute the global-averaged spherical harmonics over the nearest-neighbor bonds of a set of particles. Returns the raw (complex) spherical harmonic values.

Parameters

- **neigh_min** – Minimum number of neighbor environment sizes to consider
- **neigh_max** – Maximum number of neighbor environment sizes to consider (inclusive)
- **lmax** – Maximum spherical harmonic degree l
- **negative_m** – Include negative m spherical harmonics in the output array?
- **reference_frame** – 'neighborhood': use diagonal inertia tensor reference frame; 'particle_local': use the given orientations array; 'global': do not rotate
- **orientations** – Per-particle orientations, only used when reference_frame == 'particle_local'
- **rmax_guess** – Initial guess of the distance to find *neigh_max* nearest neighbors. Only affects algorithm speed.
- **noise_samples** – Number of random noisy samples of positions to average the result over (disabled if 0)
- **noise_magnitude** – Magnitude of (normally-distributed) noise to apply to noise_samples different positions (disabled if *noise_samples == 0*)
- **nlist** – Freud neighbor list object to use (*None* to compute for neighbors up to *neigh_max*)

This function uses the following citations:

```
@misc{freud2016,
        title = {freud},
        url = {https://doi.org/10.5281/zenodo.166564},
        abstract = {First official open-source release, includes a zenodo DOI for␣
↪citations.},
        author = {Harper, Eric S and Spellings, Matthew and Anderson, Joshua A␣
↪and Glotzer, Sharon C},
        month = nov,
        year = {2016},
        doi = {10.5281/zenodo.166564},
}

@article{spellings2018,
        title = {Machine learning for crystal identification and discovery},
        volume = {64},
        url = {https://dx.doi.org/10.1002/aic.16157},
        doi = {10.1002/aic.16157},
        number = {6},
        journal = {AIChE Journal},
        author = {Spellings, Matthew and Glotzer, Sharon C},
        year = {2018},
        pages = {2198--2206},
}
```

6.3 Voronoi Descriptors

This module computes descriptors based on the Voronoi tessellation of the system.

pythia.voronoi.**angle_histogram**(*box*, *positions*, *bins*, *buffer_distance=None*, *area_weight_mode='product'*)

Compute the area-weighted (a_i + a_j) angle histogram of all pairs of faces for the voronoi polyhedron of each particle. Sums the areas into the given number of bins (from 0 to pi).

Parameters

- **bins** – Number of bins to use for the histogram
- **buffer_distance** – Distance to copy parts of the simulation box for periodic boundary conditions in the voronoi diagram computation
- **area_weight_mode** – Whether the weight for each pair of faces should be the sum ('sum') or product ('product') of the face areas

This function uses the following citations:

```
@misc{freud2016,
        title = {freud},
        url = {https://doi.org/10.5281/zenodo.166564},
        abstract = {First official open-source release, includes a zenodo DOI for␣
↪citations.},
        author = {Harper, Eric S and Spellings, Matthew and Anderson, Joshua A␣
↪and Glotzer, Sharon C},
        month = nov,
        year = {2016},
        doi = {10.5281/zenodo.166564},
}
```

6.4 Learned Representations

The *pythia.learned* module implements building blocks for creating learned representations of particle systems. They are implemented within the keras framework for ease of integration into standard workflows.

class pythia.learned.bonds.**InertiaRotation**(*num_rotations=1*, *initial_mass_variance=0.25*, *center=False*, ***kwargs*)

Generate rotation-invariant point clouds by orienting via principal axes of inertia

InertiaRotation takes an array of neighborhood points (shape *(..., num_neighbors, 3)*) and outputs one or more copies which have been rotated according to the principal axes of inertia of the neighborhood. It does this using masses that can be varied for each point and each rotation.

For an input of shape *(..., num_neighbors, 3)*, *InertiaRotation* produces an output of shape *(..., num_rotations, num_neighbors, 3)*.

Before computing the inertia tensor, points can optionally be centered via the *center* argument. A value of *True* centers the points as if all masses were 1, a value of *"com"* centers the points using the learned masses, and a value of *False* (the default) does not center at all.

Parameters

- **num_rotations** – number of rotations to produce
- **initial_mass_variance** – Variance of the initial mass distribution (mean 1)
- **center** – Center the mass points before computing the inertia tensor (see description above)

class pythia.learned.spherical_harmonics.**SphericalHarmonics**(*lmax*, *negative_m=False*, ***kwargs*)

Compute the (complex) spherical harmonic decomposition given a set of cartesian coordinates

For an input of shape *(..., 3)*, *SphericalHarmonics* will produce an output of shape *(..., num_sphs)*, where *num_sphs* is the number of spherical harmonics produced given the *lmax* and *negative_m* arguments.

Parameters

- **lmax** – maximum spherical harmonic degree to compute
- **negative_m** – If True, consider m=-l to m=l for each spherical harmonic l; otherwise, consider m=0 to m=l

class pythia.learned.spherical_harmonics.**NeighborAverage**(**args*, ***kwargs*)

Compute a weighted average an array of complex-valued spherical harmonics over all points in a neighborhood

Given an input of shape *(..., num_rotations, num_neighbors, num_sphs)*, *NeighborAverage* will produce an output of shape *(..., num_rotations, num_sphs)*. Each neighbor from each rotation is assigned a learnable weight to contribute to the sum.

class pythia.learned.spherical_harmonics.**ComplexProjection**(*num_projections=1*, *conversion='abs'*, *activation=None*, *kernel_initializer='glorot_uniform'*, *bias_initializer='random_normal'*, ***kwargs*)

Compute one or more linear projections of a complex-valued function

Given an input of shape *(..., num_rotations, num_sphs)*, *ComplexProjection* produces an output of shape *(.., num_rotations, num_projections)*.

Outputs are converted to real numbers by taking the absolute value of the projection output by default, but the real or imaginary components can also be taken instead.

Parameters

- **num_projections** – Number of projections (i.e. number of neurons) to create for each rotation
- **conversion** – Method to make the projection output real: 'abs' (absolute value), 'angle' (complex phase), 'real' (real component), 'imag' (imaginary component), or a comma-separated list of these values (i.e. 'real,imag')
- **activation** – Keras activation function for the layer
- **kernel_initializer** – Keras initializer for the projection weights matrix
- **bias_initializer** – Keras initializer for the projection bias matrix

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index