PairwiseDistance

class **gtda.diagrams.PairwiseDistance**(*metric='landscape'*, *metric_params=None*, *order=2.0*, *n_jobs=None*) [source]

Distances between pairs of persistence diagrams.

Given two collections of persistence diagrams consisting of birth-death-dimension triples [b, d, q], a collection of distance matrices or a single distance matrix between pairs of diagrams is calculated according to the following steps:

1. All diagrams are partitioned into subdiagrams corresponding to distinct homology dimensions.
2. Pairwise distances between subdiagrams of equal homology dimension are calculated according to the parameters *metric* and *metric_params*. This gives a collection of distance matrices, $\mathbf{D} = (D_{q_1}, \ldots, D_{q_k})$.
3. The final result is either $\mathbf{D}$ itself as a three-dimensional array, or a single distance matrix constructed by taking norms of the vectors of distances between diagram pairs.

Important notes:

- Input collections of persistence diagrams for this transformer must satisfy certain requirements, see e.g. `fit`.
- The shape of outputs of `transform` depends on the value of the *order* parameter.

Parameters

- **metric** (`'bottleneck'` | `'wasserstein'` | `'betti'` | `'landscape'` | `'silhouette'` | `'heat'` | `'persistence_image'`, optional, default: `'landscape'`) –

  Distance or dissimilarity function between subdiagrams:

  - `'bottleneck'` and `'wasserstein'` refer to the identically named perfect-matching–based notions of distance.
  - `'betti'` refers to the L^p distance between Betti curves.
  - `'landscape'` refers to the L^p distance between persistence landscapes.
  - `'silhouette'` refers to the L^p distance between silhouettes.
  - `'heat'` refers to the L^p distance between Gaussian-smoothed diagrams.
  - `'persistence_image'` refers to the L^p distance between Gaussian-smoothed diagrams represented on birth-persistence axes.

- **metric_params** (dict or None, optional, default: `None`) –

  Additional keyword arguments for the metric function (passing `None` is equivalent to passing the defaults described below):

  - If `metric == 'bottleneck'` the only argument is *delta* (float, default: `0.01`). When equal to `0.` , an exact algorithm is used; otherwise, a faster approximate algorithm is used and symmetry is not guaranteed.
  - If `metric == 'wasserstein'` the available arguments are *p* (float, default: `2.`) and *delta* (float, default: `0.01`). Unlike the case of `'bottleneck'` , *delta* cannot be set to `0.` and an exact algorithm is not available.
  - If `metric == 'betti'` the available arguments are *p* (float, default: `2.`) and *n_bins* (int, default: `100`).
  - If `metric == 'landscape'` the available arguments are *p* (float, default: `2.`), *n_bins* (int, default: `100`) and *n_layers* (int, default: `1`).
  - If `metric == 'silhouette'` the available arguments are *p* (float, default: `2.`), *power* (float, default: `1.`) and *n_bins* (int, default: `100`).
  - If `metric == 'heat'` the available arguments are *p* (float, default: `2.`), *sigma* (float, default: `0.1`) and *n_bins* (int, default: `100`).
  - If `metric == 'persistence_image'` the available arguments are *p* (float, default: `2.`), *sigma* (float, default: `0.1`), *n_bins* (int, default: `100`) and *weight_function* (callable or None, default: `None`).

- **order** (float or None, optional, default: `2.`) – If `None` , `transform` returns for each pair of diagrams a vector of distances corresponding to the dimensions in `homology_dimensions_` . Otherwise, the p-norm of these vectors with p equal to *order* is taken.
- **n_jobs** (int or None, optional, default: `None`) – The number of jobs to use for the computation. `None` means 1 unless in a `joblib.parallel_backend` context. `-1` means using all processors.

`effective_metric_params_`

Dictionary containing all information present in *metric_params* as well as relevant quantities computed in `fit` .

Type

dict

`homology_dimensions_`

Homology dimensions seen in `fit` , sorted in ascending order.

Type

tuple

See also

`Amplitude` , `Scaler` , `Filtering` , `DettiCurve` , `PersistenceLandscape` , `PersistenceImage` , `HeatKernel` , `Silhouette` , `gtda.homology.VietorisRipsPersistence`

Notes

To compute distances without first splitting the computation between different homology dimensions, data should be first transformed by an instance of `ForgetDimension` .

Hera is used as a C++ backend for computing bottleneck and Wasserstein distances between persistence diagrams. Python bindings were modified for performance from the Dyonisus 2 package.

`__init__`(*metric='landscape'*, *metric_params=None*, *order=2.0*, *n_jobs=None*) [source]

Initialize self. See help(type(self)) for accurate signature.

`fit`(*X*, *y=None*) [source]

Store all observed homology dimensions in `homology_dimensions_` and compute `effective_metric_params_` . Then, return the estimator.

This method is here to implement the usual scikit-learn API and hence work in pipelines.

Parameters

- **X** (*ndarray of shape (n_samples_fit, n_features, 3)*) – Input data. Array of persistence diagrams, each a collection of triples [b, d, q] representing persistent topological features through their birth (b), death (d) and homology dimension (q). It is important that, for each possible homology dimension, the number of triples for which q equals that homology dimension is constants across the entries of *X*.
- **y** (*None*) – There is no need for a target in a transformer, yet the pipeline API requires this parameter.

Returns

self

Return type

object

`fit_transform`(*X*, *y=None*, ***fit_params*)

Fit to data, then transform it.

Fits transformer to *X* and *y* with optional parameters *fit_params* and returns a transformed version of *X*.

Parameters

- **X** (*ndarray of shape (n_samples_fit, n_features, 3)*) – Input data. Array of persistence diagrams, each a collection of triples [b, d, q] representing persistent topological features through their birth (b), death (d) and homology dimension (q). It is important that, for each possible homology dimension, the number of triples for which q equals that homology dimension is constants across the entries of *X*.
- **y** (*None*) – There is no need for a target in a transformer, yet the pipeline API requires this parameter.

Returns

Xt – Distance matrix or collection of distance matrices between diagrams in *X* and diagrams seen in `fit` . In the second case, index i along axis 2 corresponds to the i-th homology dimension in `homology_dimensions_` .

Return type

ndarray of shape (n_samples, n_samples_fit, n_homology_dimensions) if *order* is `None` , else (n_samples, n_samples_fit)

`get_params`(*deep=True*)

Get parameters for this estimator.

Parameters

deep (*bool, default=True*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns

params – Parameter names mapped to their values.

Return type

mapping of string to any

`set_params`(***params*)

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as pipelines). The latter have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

Parameters

****params** (*dict*) – Estimator parameters.

Returns

self – Estimator instance.

Return type

object

`transform`(*X*, *y=None*) [source]

Computes a distance or vector of distances between the diagrams in *X* and the diagrams seen in `fit` .

Parameters

- **X** (*ndarray of shape (n_samples, n_features, 3)*) – Input data. Array of persistence diagrams, each a collection of triples [b, d, q] representing persistent topological features through their birth (b), death (d) and homology dimension (q). It is important that, for each possible homology dimension, the number of triples for which q equals that homology dimension is constants across the entries of *X*.
- **y** (*None*) – There is no need for a target in a transformer, yet the pipeline API requires this parameter.

Returns

Xt – Distance matrix or collection of distance matrices between diagrams in *X* and diagrams seen in `fit` . In the second case, index i along axis 2 corresponds to the i-th homology dimension in `homology_dimensions_` .

Return type

ndarray of shape (n_samples, n_samples_fit, n_homology_dimensions) if *order* is `None` , else (n_samples, n_samples_fit)