sklearn.metrics.DistanceMetric

class sklearn.metrics.**DistanceMetric**

DistanceMetric class

This class provides a uniform interface to fast distance metric functions. The various metrics can be accessed via the get_metric class method and the metric string identifier (see below).

Examples

```
>>> from sklearn.metrics import DistanceMetric
>>> dist = DistanceMetric.get_metric('euclidean')
>>> X = [[0, 1, 2],
         [3, 4, 5]]
>>> dist.pairwise(X)
array([[ 0.        ,  5.19615242],
       [ 5.19615242,  0.        ]])
```

Available Metrics

The following lists the string metric identifiers and the associated distance metric classes:

Metrics intended for real-valued vector spaces:

identifier	class name	args	distance function
“euclidean”	EuclideanDistance	•	sqrt(sum((x - y)^2))
“manhattan”	ManhattanDistance	•	sum(|x - y|)
“chebyshev”	ChebyshevDistance	•	max(|x - y|)
“minkowski”	MinkowskiDistance	p, w	sum(w * |x - y|^p)^(1/p)
“wminkowski”	WMinkowskiDistance	p, w	sum(|w * (x - y)|^p)^(1/p)
“seuclidean”	SEuclideanDistance	V	sqrt(sum((x - y)^2 / V))
“mahalanobis”	MahalanobisDistance	V or VI	sqrt((x - y)' V^-1 (x - y))

Deprecated since version 1.1: WMinkowskiDistance is deprecated in version 1.1 and will be removed in version 1.3. Use MinkowskiDistance instead. Note that in MinkowskiDistance, the weights are applied to the absolute differences already raised to the p power. This is different from WMinkowskiDistance where weights are applied to the absolute differences before raising to the p power. The deprecation aims to remain consistent with SciPy 1.8 convention.

Metrics intended for two-dimensional vector spaces: Note that the haversine distance metric requires data in the form of [latitude, longitude] and both inputs and outputs are in units of radians.

identifier	class name	distance function
“haversine”	HaversineDistance	2 arcsin(sqrt(sin^2(0.5*dx) + cos(x1)cos(x2)sin^2(0.5*dy)))

Metrics intended for integer-valued vector spaces: Though intended for integer-valued vectors, these are also valid metrics in the case of real-valued vectors.

identifier	class name	distance function
“hamming”	HammingDistance	N_unequal(x, y) / N_tot
“canberra”	CanberraDistance	sum(|x - y| / (|x| + |y|))
“braycurtis”	BrayCurtisDistance	sum(|x - y|) / (sum(|x|) + sum(|y|))

Metrics intended for boolean-valued vector spaces: Any nonzero entry is evaluated to “True”. In the listings below, the following abbreviations are used:

- N : number of dimensions
- NTT : number of dims in which both values are True
- NTF : number of dims in which the first value is True, second is False
- NFT : number of dims in which the first value is False, second is True
- NFF : number of dims in which both values are False
- NNEQ : number of non-equal dimensions, NNEQ = NTF + NFT
- NNZ : number of nonzero dimensions, NNZ = NTF + NFT + NTT

Toggle Menu

identifier	class name	distance function
“jaccard”	JaccardDistance	NNEQ / NNZ
“matching”	MatchingDistance	NNEQ / N
“dice”	DiceDistance	NNEQ / (NTT + NNZ)
“kulsinski”	KulsinskiDistance	(NNEQ + N - NTT) / (NNEQ + N)
“rogerstanimoto”	RogersTanimotoDistance	2 * NNEQ / (N + NNEQ)
“russellrao”	RussellRaoDistance	(N - NTT) / N
“sokalmichener”	SokalMichenerDistance	2 * NNEQ / (N + NNEQ)
“sokalsneath”	SokalSneathDistance	NNEQ / (NNEQ + 0.5 * NTT)

User-defined distance:

identifier	class name	args
“pyfunc”	PyFuncDistance	func

Here `func` is a function which takes two one-dimensional numpy arrays, and returns a distance. Note that in order to be used within the BallTree, the distance must be a true metric: i.e. it must satisfy the following properties

1. Non-negativity: d(x, y) >= 0
2. Identity: d(x, y) = 0 if and only if x == y
3. Symmetry: d(x, y) = d(y, x)
4. Triangle Inequality: d(x, y) + d(y, z) >= d(x, z)

Because of the Python object overhead involved in calling the python function, this will be fairly slow, but it will have the same scaling as other distances.

Methods

`dist_to_rdist`	Convert the true distance to the rank-preserving surrogate distance.
`get_metric`	Get the given distance metric from the string identifier.
`pairwise`	Compute the pairwise distances between X and Y
`rdist_to_dist`	Convert the rank-preserving surrogate distance to the distance.

`dist_to_rdist`()

Convert the true distance to the rank-preserving surrogate distance.

The surrogate distance is any measure that yields the same rank as the distance, but is more efficient to compute. For example, the rank-preserving surrogate distance of the Euclidean metric is the squared-euclidean distance.

Parameters:

dist : *double*
True distance.

Returns:

double
Surrogate distance.

`get_metric`()

Get the given distance metric from the string identifier.

See the docstring of DistanceMetric for a list of available metrics.

Parameters:

metric : *str or class name*
The distance metric to use

****kwargs**
additional arguments will be passed to the requested metric

`pairwise`()

Compute the pairwise distances between X and Y

This is a convenience routine for the sake of testing. For many metrics, the utilities in scipy.spatial.distance.cdist and

Toggle Menu

scipy.spatial.distance.pdist will be faster.

Parameters:

X : *array-like*
Array of shape (Nx, D), representing Nx points in D dimensions.

Y : *array-like (optional)*
Array of shape (Ny, D), representing Ny points in D dimensions. If not specified, then Y=X.

Returns:

dist : *ndarray*
The shape (Nx, Ny) array of pairwise distances between points in X and Y.

`rdist_to_dist`()

Convert the rank-preserving surrogate distance to the distance.

The surrogate distance is any measure that yields the same rank as the distance, but is more efficient to compute. For example, the rank-preserving surrogate distance of the Euclidean metric is the squared-euclidean distance.

Parameters:

rdist : *double*
Surrogate distance.

Returns:

double
True distance.