dask-distance Documentation

Release 0.2.0+0.g0b36556.dirty

John Kirkham

Oct 09, 2017

Contents

Contents:

CHAPTER 1

dask-distance

Distance computations with Dask (akin to scipy.spatial.distance)

- Free software: BSD 3-Clause
- Documentation: https://dask-distance.readthedocs.io.

1.1 Features

- TODO

1.2 Credits

This package was created with Cookiecutter and the dask-image/dask-image-cookiecutter project template.

CHAPTER 2

Installation

2.1 Stable release

To install dask-distance, run this command in your terminal:

```
$ pip install dask-distance
```

This is the preferred method to install dask-distance, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for dask-distance can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/jakirkham/dask-distance
```

Or download the tarball:

```
$ curl  -OL https://github.com/jakirkham/dask-distance/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use dask-distance in a project:

```
import dask_distance
```

CHAPTER 4

API

4.1 dask_distance package

`dask_distance.`**`braycurtis`**(*u*, *v*)

Finds the Bray-Curtis distance between two 1-D arrays.

$$\frac{\sum_i |u_i - v_i|}{\sum_i |u_i + v_i|}$$

Parameters

- **u** – 1-D array or collection of 1-D arrays
- **v** – 1-D array or collection of 1-D arrays

Returns Bray-Curtis distance

Return type float

`dask_distance.`**`canberra`**(*u*, *v*)

Finds the Canberra distance between two 1-D arrays.

$$\sum_i \frac{|u_i - v_i|}{|u_i| + |v_i|}$$

Parameters

- **u** – 1-D array or collection of 1-D arrays
- **v** – 1-D array or collection of 1-D arrays

Returns Canberra distance

Return type float

`dask_distance.`**`cdist`**(*XA*, *XB*, *metric=u'euclidean'*, ***kwargs*)

Finds the distance matrix using the metric on each pair of points.

Parameters

- **XA** – 2-D array of points
- **XB** – 2-D array of points
- **metric** – string or callable
- ****kwargs** – provided to the metric (see below)

Keyword Arguments

- **p** – p-norm for minkowski only (default: 2)
- **V** – 1-D array of variances for seuclidean only (default: estimated from XA and XB)
- **VI** – Inverse of the covariance matrix for mahalanobis only (default: estimated from XA and XB)
- **w** – 1-D array of weights for wminkowski only (required)

Returns distance between each combination of points

Return type array

`dask_distance.`**`chebyshev`**`(u, v)`

Finds the Chebyshev distance between two 1-D arrays.

$$\max_i |u_i - v_i|$$

Parameters

- **u** – 1-D array or collection of 1-D arrays
- **v** – 1-D array or collection of 1-D arrays

Returns Chebyshev distance

Return type float

`dask_distance.`**`cityblock`**`(u, v)`

Finds the City Block (Manhattan) distance between two 1-D arrays.

$$\sum_i |u_i - v_i|$$

Parameters

- **u** – 1-D array or collection of 1-D arrays
- **v** – 1-D array or collection of 1-D arrays

Returns City Block (Manhattan) distance

Return type float

`dask_distance.`**`correlation`**`(u, v)`

Finds the correlation distance between two 1-D arrays.

$$1 - \frac{(u - \bar{u}) \cdot (v - \bar{v})}{\|u - \bar{u}\|_2 \|v - \bar{v}\|_2}$$

Parameters

- **u** – 1-D array or collection of 1-D arrays
- **v** – 1-D array or collection of 1-D arrays

Returns correlation distance

Return type float

`dask_distance.cosine(u, v)`

Finds the Cosine distance between two 1-D arrays.

$$1 - \frac{u \cdot v}{\|u\|_2 \|v\|_2}$$

Parameters

- **u** – 1-D array or collection of 1-D arrays
- **v** – 1-D array or collection of 1-D arrays

Returns Cosine distance

Return type float

`dask_distance.dice(u, v)`

Finds the Dice dissimilarity between two 1-D bool arrays.

$$\frac{c_{TF} + c_{FT}}{2 \cdot c_{TT} + c_{TF} + c_{FT}}$$

where $c_{XY} = \sum_i \delta_{u_i X} \delta_{v_i Y}$

Parameters

- **u** – 1-D bool array or collection of 1-D bool arrays
- **v** – 1-D bool array or collection of 1-D bool arrays

Returns Dice dissimilarity

Return type float

`dask_distance.euclidean(u, v)`

Finds the Euclidean distance between two 1-D arrays.

$$\|u - v\|_2$$

Parameters

- **u** – 1-D array or collection of 1-D arrays
- **v** – 1-D array or collection of 1-D arrays

Returns Euclidean distance

Return type float

`dask_distance.hamming(u, v)`

Finds the Hamming distance between two 1-D bool arrays.

$$\frac{c_{TF} + c_{FT}}{c_{TT} + c_{TF} + c_{FT} + c_{FF}}$$

where $c_{XY} = \sum_i \delta_{u_i X} \delta_{v_i Y}$

Parameters

- **u** – 1-D bool array or collection of 1-D bool arrays
- **v** – 1-D bool array or collection of 1-D bool arrays

Returns Hamming distance

Return type float

`dask_distance.`**`jaccard`**(*u*, *v*)

Finds the Jaccard-Needham dissimilarity between two 1-D bool arrays.

$$\frac{c_{TF} + c_{FT}}{c_{TT} + c_{TF} + c_{FT}}$$

where $c_{XY} = \sum_i \delta_{u_i X} \delta_{v_i Y}$

Parameters

- **u** – 1-D bool array or collection of 1-D bool arrays
- **v** – 1-D bool array or collection of 1-D bool arrays

Returns Jaccard-Needham dissimilarity

Return type float

`dask_distance.`**`kulsinski`**(*u*, *v*)

Finds the Kulsinski dissimilarity between two 1-D bool arrays.

$$\frac{2 \cdot (c_{TF} + c_{FT}) + c_{FF}}{c_{TT} + 2 \cdot (c_{TF} + c_{FT}) + c_{FF}}$$

where $c_{XY} = \sum_i \delta_{u_i X} \delta_{v_i Y}$

Parameters

- **u** – 1-D bool array or collection of 1-D bool arrays
- **v** – 1-D bool array or collection of 1-D bool arrays

Returns Kulsinski dissimilarity

Return type float

`dask_distance.`**`mahalanobis`**(*u*, *v*, *VI*)

Finds the Mahalanobis distance between two 1-D arrays.

$$\sqrt{(u - v) \cdot V^{-1} \cdot (u - v)^T}$$

Parameters

- **u** – 1-D array or collection of 1-D arrays
- **v** – 1-D array or collection of 1-D arrays
- **VI** – Inverse of the covariance matrix

Returns Mahalanobis distance

Return type float

`dask_distance.`**`minkowski`**(*u*, *v*, *p*)

Finds the Minkowski distance between two 1-D arrays.

$$\left(\sum_i |u_i - v_i|^p \right)^{\frac{1}{p}}$$

Parameters

- **u** – 1-D array or collection of 1-D arrays
- **v** – 1-D array or collection of 1-D arrays
- **p** – degree of the norm to use

Returns Minkowski distance

Return type float

`dask_distance.`**`pdist`**(*X*, *metric=u'euclidean'*, ***kwargs*)

Finds the pairwise condensed distance matrix using the metric.

Parameters

- **`X`** – 2-D array of points
- **`metric`** – string or callable
- **`**kwargs`** – provided to the metric (see below)

Keyword Arguments

- **`p`** – p-norm for minkowski only (default: 2)
- **`V`** – 1-D array of variances for seuclidean only (default: estimated from X)
- **`VI`** – Inverse of the covariance matrix for mahalanobis only (default: estimated from X)
- **`w`** – 1-D array of weights for wminkowski only (required)

Returns condensed distance between each pair

Return type array

Note: Tries to avoid redundant computations as much as possible. However this is limited in its ability to do this based on the chunk size of X (particularly along the first dimension). Smaller chunks will increase savings though there may be other tradeoffs.

`dask_distance.`**`rogerstanimoto`**(*u*, *v*)

Finds the Rogers-Tanimoto dissimilarity between two 1-D bool arrays.

$$\frac{2 \cdot (c_{TF} + c_{FT})}{c_{TT} + 2 \cdot (c_{TF} + c_{FT}) + c_{FF}}$$

where $c_{XY} = \sum_i \delta_{u_i X} \delta_{v_i Y}$

Parameters

- **`u`** – 1-D bool array or collection of 1-D bool arrays
- **`v`** – 1-D bool array or collection of 1-D bool arrays

Returns Rogers-Tanimoto dissimilarity

Return type float

`dask_distance.`**`russellrao`**(*u*, *v*)

Finds the Russell-Rao dissimilarity between two 1-D bool arrays.

$$\frac{c_{TF} + c_{FT} + c_{FF}}{c_{TT} + c_{TF} + c_{FT} + c_{FF}}$$

where $c_{XY} = \sum_i \delta_{u_i X} \delta_{v_i Y}$

Parameters

- **`u`** – 1-D bool array or collection of 1-D bool arrays
- **`v`** – 1-D bool array or collection of 1-D bool arrays

Returns Russell-Rao dissimilarity

Return type float

`dask_distance.`**`seuclidean`**(*u*, *v*, *V*)

Finds the standardized Euclidean distance between two 1-D arrays.

$$\sqrt{\sum_i \left(\frac{(u_i - v_i)^2}{V_i}\right)}$$

Parameters

- **u** – 1-D array or collection of 1-D arrays
- **v** – 1-D array or collection of 1-D arrays
- **V** – 1-D array of variances

Returns standardized Euclidean

Return type float

`dask_distance.`**`sokalmichener`**(*u*, *v*)

Finds the Sokal-Michener dissimilarity between two 1-D bool arrays.

$$\frac{2 \cdot (c_{TF} + c_{FT})}{c_{TT} + 2 \cdot (c_{TF} + c_{FT}) + c_{FF}}$$

where $c_{XY} = \sum_i \delta_{u_i X} \delta_{v_i Y}$

Parameters

- **u** – 1-D bool array or collection of 1-D bool arrays
- **v** – 1-D bool array or collection of 1-D bool arrays

Returns Sokal-Michener dissimilarity

Return type float

`dask_distance.`**`sokalsneath`**(*u*, *v*)

Finds the Sokal-Sneath dissimilarity between two 1-D bool arrays.

$$\frac{2 \cdot (c_{TF} + c_{FT})}{c_{TT} + 2 \cdot (c_{TF} + c_{FT})}$$

where $c_{XY} = \sum_i \delta_{u_i X} \delta_{v_i Y}$

Parameters

- **u** – 1-D bool array or collection of 1-D bool arrays
- **v** – 1-D bool array or collection of 1-D bool arrays

Returns Sokal-Sneath dissimilarity

Return type float

`dask_distance.`**`sqeuclidean`**(*u*, *v*)

Finds the squared Euclidean distance between two 1-D arrays.

$$\|u - v\|_2^2$$

Parameters

- **u** – 1-D array or collection of 1-D arrays

- **v** – 1-D array or collection of 1-D arrays

Returns squared Euclidean distance

Return type float

`dask_distance.`**`squareform`**(*X*, *force=u'no'*)

Converts between dense and sparse distance matrices

Parameters

- **X** – 2-D square symmetric matrix or 1-D vector of distances
- **force** – whether to force to a vector or a matrix

Returns 1-D vector or 2-D square symmetric matrix of distances

Return type array

`dask_distance.`**`wminkowski`**(*u*, *v*, *p*, *w*)

Finds the weighted Minkowski distance between two 1-D arrays.

$$\left(\sum_{i}|w_i \cdot (u_i - v_i)|^p\right)^{\frac{1}{p}}$$

Parameters

- **u** – 1-D array or collection of 1-D arrays
- **v** – 1-D array or collection of 1-D arrays
- **p** – degree of the norm to use
- **w** – 1-D array of weights

Returns Minkowski distance

Return type float

`dask_distance.`**`yule`**(*u*, *v*)

Finds the Yule dissimilarity between two 1-D bool arrays.

$$\frac{2 \cdot c_{TF} \cdot c_{FT}}{c_{TT} \cdot c_{FF} + c_{TF} \cdot c_{FT}}$$

where $c_{XY} = \sum_i \delta_{u_i X} \delta_{v_i Y}$

Parameters

- **u** – 1-D bool array or collection of 1-D bool arrays
- **v** – 1-D bool array or collection of 1-D bool arrays

Returns Yule dissimilarity

Return type float

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/jakirkham/dask-distance/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

dask-distance could always use more documentation, whether as part of the official dask-distance docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/jakirkham/dask-distance/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *dask-distance* for local development.

1. Fork the *dask-distance* repo on GitHub.
2. Clone your fork locally:

   ```
   $ git clone git@github.com:your_name_here/dask-distance.git
   ```

3. Install your local copy into an environment. Assuming you have conda installed, this is how you set up your fork for local development (on Windows drop *source*). Replace *"<some version>"* with the Python version used for testing.:

   ```
   $ conda create -n dask-distanceenv python="<some version>"
   $ source activate dask-distanceenv
   $ python setup.py develop
   ```

4. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions:

   ```
   $ flake8 dask_distance tests
   $ python setup.py test or py.test
   ```

   To get flake8, just conda install it into your environment.

6. Commit your changes and push your branch to GitHub:

   ```
   $ git add .
   $ git commit -m "Your detailed description of your changes."
   $ git push origin name-of-your-bugfix-or-feature
   ```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5, and 3.6. Check https://travis-ci.org/jakirkham/dask-distance/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests/test_dask_distance.py
```

CHAPTER 6

Credits

6.1 Development Lead

- John Kirkham, Howard Hughes Medical Institute <kirkhamj@janelia.hhmi.org>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index