ristretto Documentation

Release 0.1.2

N. Benjamin Erichson

Apr 17, 2021

Contents:

The idea of randomized low-rank matrix approximations is to restrict the high-dimensional input data matrix to a low-dimensional space. In plain words, the aim is to find a smaller matrix which captures the essential information of the input matrix. This smaller matrix can then be used to extract (learn) the coherent structure of the data. Probabilistic algorithms considerably reduce the computational demands of traditional (deterministic) algorithms, and the computational advantage becomes pronounced with increasing matrix dimensions.

The Python software library ristretto provides a collection of randomized matrix algorithms which can be used for dimension reduction. Overview of implemented routines:

- Randomized singular value decomposition:`from ristretto.svd import compute_rsvd`.
- Randomized interpolative decomposition:`from ristretto.interp_decomp import compute_rinterp_decomp`.
- Randomized CUR decomposition: `from ristretto.cur import compute_rcur`.
- Randomized LU decompositoin: `from ristretto.lu import compute_rlu`.
- Randomized nonnegative matrix factorization: `from ristretto.nmf import compute_rnmf_fhals`.

CHAPTER 1

Get started

1.1 Obtaining the Latest Software via GIT

To get the latest stable and development versions of ristretto run:

```
$ git clone https://github.com/erichson/ristretto
```

Then, to build and install the package, run from within the main directory in the release:

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

Note you will need the following 3rd party packages installed in your environment:

- numpy
- scipy
- Cython
- scikit-learn
- nose

After successfully installing the ristretto library, the unit tests can be run by:

```
$ python setup.py test
```

CHAPTER 2

References

- N. Benjamin Erichson, et al. 'Randomized Matrix Decompositions using R.' (2016)
- Sergey Voronin, Per-Gunnar Martinsson. 'RSVDPACK: Subroutines for computing partial singular value decompositions via randomized sampling on single core, multi core, and GPU architectures.' (2015)
- Nathan Halko, et al. 'Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions.' (2011)

2.1 API Reference

This is the reference for the functions contained in ristretto.

2.1.1 `ristretto.cur`: CUR Decomposition

CUR-ID

`cur.cur`
`cur.rcur`

2.1.2 `ristretto.dmd`: DMD Decomposition

Dynamic Mode Decomposition (DMD).

`dmd.dmd`
`dmd.rdmd`

2.1.3 ristretto.eigen: Eigenvalue Decompositions

Randomized Singular Value Decomposition

eigen.reigh
eigen.reigh_nystroem
eigen.reigh_nystroem_col

2.1.4 ristretto.interp_decomp: Interpolation Decompositions

Interpolative decomposition (ID)

interp_decomp.interp_decomp
interp_decomp.rinterp_decomp

2.1.5 ristretto.lu: LU Decomposition

Randomized LU Decomposition

lu.rlu

2.1.6 ristretto.nmf: Non-negative Matrix Factorization

nmf.nmf
nmf.rnmf

2.1.7 ristretto.pca: Principal Component Analysis

Principal Component Analysis (PCA).

pca.robspca
pca.rspca
pca.spca

2.1.8 ristretto.qb: QB Decomposition

Randomized QB Decomposition

qb.rqb

2.1.9 ristretto.svd: SVD Decomposition

Random Singular Value Decomposition.

svd.rsvd	

2.1.10 ristretto.utils: Utility Functions

Utility Functions.

utils.check_non_negative	
utils.check_random_state	
utils.conjugate_transpose(A)	Performs conjugate transpose of A
utils.safe_sparse_dot	
utils.nmf_data(m, n, k[, factor_type, ...])	

ristretto.utils.conjugate_transpose

ristretto.utils.**conjugate_transpose**(*A*)

Performs conjugate transpose of A

ristretto.utils.nmf_data

ristretto.utils.**nmf_data**(*m*, *n*, *k*, *factor_type='normal'*, *noise_type='normal'*, *noiselevel=0*)

2.1.11 ristretto.sketch: Sketching related functions

Transforms

Functions for approximating the range of a matrix A.

sketch.transforms.randomized_uniform_sampling(A, l)	Uniform randomized sampling transform.
sketch.transforms.johnson_lindenstrauss(A, l)	Given an m x n matrix A, and an integer l, this scheme computes an m x l orthonormal matrix Q whose range approximates the range of A
sketch.transforms.sparse_johnson_lindenstrauss(A, l)	Given an m x n matrix A, and an integer l, this scheme computes an m x l orthonormal matrix Q whose range approximates the range of A
sketch.transforms.fast_johnson_lindenstrauss(A, l)	Given an m x n matrix A, and an integer l, this scheme computes an m x l orthonormal matrix Q whose range approximates the range of A

ristretto.sketch.transforms.randomized_uniform_sampling

ristretto.sketch.transforms.**randomized_uniform_sampling**(*A*, *l*, *axis=1*, *random_state=None*)

Uniform randomized sampling transform.

Given an m x n matrix A, and an integer l, this returns an m x l random subset of the range of A.

ristretto.sketch.transforms.johnson_lindenstrauss

ristretto.sketch.transforms.**johnson_lindenstrauss**(*A*, *l*, *axis=1*, *random_state=None*)

Given an m x n matrix A, and an integer l, this scheme computes an m x l orthonormal matrix Q whose range approximates the range of A

ristretto.sketch.transforms.sparse_johnson_lindenstrauss

ristretto.sketch.transforms.**sparse_johnson_lindenstrauss**(*A*, *l*, *density=None*, *axis=1*, *random_state=None*)

Given an m x n matrix A, and an integer l, this scheme computes an m x l orthonormal matrix Q whose range approximates the range of A

density : sparse matrix density

ristretto.sketch.transforms.fast_johnson_lindenstrauss

ristretto.sketch.transforms.**fast_johnson_lindenstrauss**(*A*, *l*, *axis=1*, *random_state=None*)

Given an m x n matrix A, and an integer l, this scheme computes an m x l orthonormal matrix Q whose range approximates the range of A

Utility Functions

Module containing utility functions for

sketch.utils.orthonormalize(A[, ...])	orthonormalize the columns of A via QR decomposition
sketch.utils.perform_subspace_iterations(A, Q)	perform subspace iterations on Q

ristretto.sketch.utils.orthonormalize

ristretto.sketch.utils.**orthonormalize**(*A*, *overwrite_a=True*, *check_finite=False*)

orthonormalize the columns of A via QR decomposition

ristretto.sketch.utils.perform_subspace_iterations

ristretto.sketch.utils.**perform_subspace_iterations**(*A*, *Q*, *n_iter=2*, *axis=1*)

perform subspace iterations on Q

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

r

Index