pyparsvd Documentation

Release 0.0.1

PyParSVD contributors

Mar 24, 2021

CONTENTS:

- The **GitHub repository** of this package can be found at PyParSVD along installation instructions, and how to get started.
- **Tutorials** can be found at PyParSVD-Tutorials
- The package uses Travis-CI for **continuous integration**.

CHAPTER ONE

SUMMARY

The *PyParSVD* library implements both a serial and a parallel singular value decomposition (SVD). The implementation of the library is conveniently: - Distributed using MPI4Py (for parallel SVD); - Streaming - data can be shown in batches to update the left singular vectors; - Randomized - further acceleration of any serial components of the overall algorithm.

The *PyParSVD* library is organized following an abstract factory design pattern, where we define a base class in *parsvd_base.py*, called *ParSVD_Base* *ParSVD base class*, that implements functions and parameters available to all derived classes. In addition, it implements two abstract functions, *initialize()* and *incorporate_data()* which implementation must be provided by the derived classes.

The classes derived from the base class are the following:

- *ParSVD_serial* (implemented in *parsvd_serial.py*) *ParSVD serial*
- *ParSVD_parallel* (implemented in *parsvd_parallel.py*) *ParSVD parallel*

These derived classes contain the actual implementation of two different different versions of SVD algorithms, one that is serial (*parsvd_serial.py*) and one that is parallel (*parsvd_parallel.py*).

It should be noted that the design pattern chosen allows for the easy addition of derived classes that can implement a new SVD versions.

The distributed (parallel) implementation of the SVD in *parsvd_parallel.py* follows (Wang et al. 2016). The streaming algorithm used in the library for both *parsvd_serial.py* and *parsvd_parallel.py* is from (Levy and Lindenbaum 1998), where the parallel QR algorithm (the TSQR method) required for the streaming feature follows (Benson et al. 2013). Finally, the randomized algorithm adopted in the library follows (Halko et al 2013).

Additionally to these modules, we also provide some post-processing capabilities to visualize the results. These are implemented in *postprocessing.py* *Postprocessing module*. The functions in post-processing can be accessed directly from the base class, and in particular from the *ParSVD object* returned by the *initialize()* and *incorporate_data()* function. They can also be accessed separately from the base class, as the post-processing module constitutes a standalone module. In practice, once you run an analysis, you can load the results at a later stage and use the post-processing module to visualize the results or you can implement you own visualization tools, that best suit your needs.

1.1 Indices and table

- genindex
- modindex

CHAPTER

TWO

PARSVD MAIN MODULES

The ParSVD main modules constitutes the backbone of the *PyParSVD* library. They are constituted by the base class:

- *ParSVD_Base* (implemented in *parsvd_base.py*) *ParSVD base class*

along with its derived classes:

- *ParSVD_serial* (implemented in *parsvd_serial.py*) *ParSVD serial*
- *ParSVD_parallel* (implemented in *parsvd_parallel.py*) *ParSVD parallel*

2.1 ParSVD base class

The **ParSVD base class** is intended to hold functions that are shared by all derived classes. It follows an abstract factory design pattern.

class ParSVD_Base(*K*, *ff*, *low_rank=False*, *results_dir='results'*)

PyParSVD base class. It implements data and methods shared across the derived classes.

Parameters

- **K** (*int*) – number of modes to truncate.
- **ff** (*int*) – forget factor.
- **low_rank** (*bool*) – if True, it uses a low rank algorithm to speed up computations.
- **results_dir** (*str*) – if specified, it saves the results in *results_dir*. Default save path is under a folder called *results* in current working path.

property K

Get the number of modes to truncate.

Returns number of modes to truncate.

Return type int

property comm

Get the parallel MPI Communicator.

Returns comm.

Return type MPI_Comm

property ff

Get the forget factor.

Returns forget factor.

Return type int

property iteration
Get the number of data incorporation performed in the streaming data ingestion.

Returns iterations.

Return type int

property low_rank
Get the low rank behaviour.

Returns low rank behaviour.

Return type bool

property modes
Get the modes.

Returns modes.

Return type ndarray

property n_modes
Get the number of modes.

Returns number of modes.

Return type int

property nprocs
Get the number processors

Returns processors.

Return type int

plot_1D_modes(*idxs=[0], title='', figsize=(12, 8), filename=None*)
See method implementation in the postprocessing module.

plot_singular_values(*idxs=[0], title='', figsize=(12, 8), filename=None*)
See method implementation in the postprocessing module.

property rank
Get the parallel MPI Rank.

Returns rank.

Return type MPI_Rank

property singular_values
Get the singular values.

Returns singular values.

Return type ndarray

2.2 ParSVD serial

class ParSVD_Serial(*K*, *ff*, *low_rank=False*, *results_dir='results'*)

PyParSVD serial class.

Parameters

- **K** (*int*) – number of modes to truncate.
- **ff** (*int*) – forget factor.
- **low_rank** (*bool*) – if True, it uses a low rank algorithm to speed up computations.
- **results_dir** (*str*) – if specified, it saves the results in *results_dir*. Default save path is under a folder called *results* in current working path.

incorporate_data(*A*)

Incorporate new data in a streaming way for SVD computation.

Parameters **A** (*ndarray*) – new data matrix.

initialize(*A*)

Initialize SVD computation with initial data.

Parameters **A** (*ndarray*) – initial data matrix.

save()

Save data.

2.3 ParSVD parallel

class ParSVD_Parallel(*K*, *ff*, *low_rank=False*, *results_dir='results'*)

PyParSVD parallel class.

Parameters

- **K** (*int*) – number of modes to truncate.
- **ff** (*int*) – forget factor.
- **low_rank** (*bool*) – if True, it uses a low rank algorithm to speed up computations.
- **results_dir** (*str*) – if specified, it saves the results in *results_dir*. Default save path is under a folder called *results* in current working path.

incorporate_data(*A*)

Incorporate new data in a streaming way for SVD computation.

Parameters **A** (*ndarray*) – new data matrix.

initialize(*A*)

Initialize SVD computation with initial data.

Parameters **A** (*ndarray*) – initial data matrix.

save()

Save data.

CHAPTER

THREE

POSTPROCESSING MODULE

The postprocessing module is intended to provide some limited support to post-process the data and results produced by **pyparsvd**. The key routines implemented are

`plot_1D_modes`(*modes*, *idxs=[0]*, *title=''*, *figsize=(12, 8)*, *path='CWD'*, *filename=None*, *rank=None*, *value='abs'*)

Plots modes of the SVD decomposition.

Parameters

- **`modes`** (*`ndarray`*) – modes.
- **`title`** (*`str`*) – if specified, title of the plot.
- **`figsize`** (*`tuple(int, int)`*) – size of the figure (width,height). Default is (12,8).
- **`path`** (*`str`*) – if specified, the plot is saved at *path*. Default is CWD.
- **`filename`** (*`str`*) – if specified, the plot is saved at *filename*. Default is None.
- **`rank`** (*`MPI_Rank`*) – MPI rank for parallel SVD.
- **`value`** (*`str`*) – whether to plot absolute or real value of modes.

`plot_singular_values`(*singular_values*, *title=''*, *figsize=(12, 8)*, *path='CWD'*, *filename=None*, *rank=None*)

Plots singular values of the SVD decomposition.

Parameters

- **`singular_values`** (*`ndarray`*) – singular values.
- **`title`** (*`str`*) – if specified, title of the plot.
- **`figsize`** (*`tuple(int, int)`*) – size of the figure (width,height). Default is (12,8).
- **`path`** (*`str`*) – if specified, the plot is saved at *path*. Default is CWD.
- **`filename`** (*`str`*) – if specified, the plot is saved at *filename*. Default is None.
- **`rank`** (*`MPI_Rank`*) – MPI rank for parallel SVD.

PYTHON MODULE INDEX

INDEX