WL-align Documentation

Release 1.0.0.post2

Matteo Frigo, Emilio Cruciani, David Coudert, Rachid Deriche, Em

Jun 09, 2021

INSTALL

wlalign is a pure Python package that implements the graph-alignment routine based on the generalization of the Weisfeiler-Lehman algorithm proposed in this paper.

The software provides the `wlalign` Python module, which includes all the **functions and tools that are necessary for computing network alignments and similarity**. In particular, specific functions are devoted to:

- Computing the **graph Jaccard index** of **similarity** between two weighted graphs.
- Solving the **graph alignment problem** with **WL-align**.

The package is available at Pypi and can be easily installed from the command line.

```
pip install wlalign
```

Talon is a free software released under MIT license.

CHAPTER

ONE

GETTING HELP

The preferred way to get assistance in running code that uses WL-align is through the issue system of the Gitlab repository where the source code is available. Developers and maintainers frequently check newly opened issues and will be happy to help you.

CHAPTER

TWO

CONTRIBUTING GUIDELINES

The development happens in the `devel` branch of the Gitlab repository, while the `master` is kept for the stable releases only. We will consider only merge requests towards the `devel` branch.

CHAPTER

THREE

HOW TO CITE

If you publish works using WL-align, please cite us as indicated here:

> Matteo Frigo, Emilio Cruciani, David Coudert, Rachid Deriche, Emanuele Natale, Samuel Deslauriers-Gauthier; Network alignment and similarity reveal atlas-based topological differences in structural connectomes. Network Neuroscience 2021; doi: https://doi.org/10.1162/netn_a_00199

In section *How to cite WL-align* you will find the Bibtex entry.

3.1 Installation

WL-align runs only on Python 3. The installation has the following dependencies:

- Numpy
- Scipy

If you are an Anaconda user, you may want to create a dedicated `wlalign-env` environment and populate it with the right dependencies, then install WL-align.

```
conda env create -n wlalign-env -f environment.yml
pip install wlalign
```

Alternatively, you can install the dependencies and WL-align all via `pip`.

```
pip install numpy
pip install scipy

pip install wlalign
```

To install WL-align directly from the source, clone this repository and run the standard local setup commands.

```
git clone https://gitlab.inria.fr/cobcom/wlalign.git
cd wlalign
pip install -U .
```

3.1.1 Check installation

To check that WL-align has been properly installed, try to import the `wlalign` module into a Python session and print the version as follows. If no error is raised, the installation has been successful.

```
>>> import wlalign
>>> print(wlalign.__version__)
```

3.1.2 For developers

If you are thinking about developing your own fork of WL-align, you may want to use the latest version in the `devel` branch of the repository and install it in editable mode.

```
git clone https://gitlab.inria.fr/cobcom/wlalign.git
cd WL-align
git checkout devel
pip install -e .
```

Tests

The package uses `unittest` as a testing suite. To run all the tests, execute the following command in the source's root directory.

```
python -m unittest -v
```

Test coverage can be checked with `coverage` as follows.

```
coverage run -m unittest
coverage report -m
```

Documentation

The sources of the documentation are in the `doc` folder. The compilation requires the `sphinx` package and the theme to be installed.

```
pip install sphinx
pip install sphinx_rtd_theme
```

To compile the documentation, move to the `doc` folder and run `make <format>`, where the format can be `html`, `latex` or any other sphinx-compatible format. To get a local copy of the the html documentation, run the `make html` command.

```
cd doc
make clean  # deletes results of previous compilations
make html
```

3.2 CLI: Command Line Interface

The user can align two graphs from the command line by providing their adjacency matrices in text form. The edge weights in each line must be separated by whitespaces.

```
wlalign --help
```

```
usage: wlalign [-h] [--first_aligned FIRST_ALIGNED] [--k INT] [--l INT]
               [--force] [-v]
               in_graph1 in_graph2 out_matching

This program uses WL-align to compute an alignment between two graphs having
the same number of nodes. It takes as input their adjacency matrices and it
returns a matching (a.k.a. alignment) between their nodes.

positional arguments:
  in_graph1             Path to the first graph to be aligned
  in_graph2             Path to the second graph to be aligned
  out_matching          Path where the matching will be saved. The first
                        element of each row is the index of the node in the
                        first graph that is aligned with the node in the
                        second graph indexed by the second element of the row

optional arguments:
  -h, --help            show this help message and exit
  --first_aligned FIRST_ALIGNED
                        The aligned version of the first input graph will be
                        saved in the specified position (default: None)
  --k INT               Width parameter (default: 2)
  --l INT               Depth parameter (default: 2)
  --force               Overwrite existing files (default: False)
  -v, --verbose         Set verbose output (default: False)
```

3.3 Alignment module

wlalign.alignment.**apply_alignment**(*graph: numpy.ndarray*, *alignment: numpy.ndarray*) → numpy.ndarray

Permute the nodes of a graph according to a specified alignment.

Parameters

- **graph** –

  np.ndarray Graph that is being permuted.

- **alignment** –

  np.ndarray n-by-2 array with one row per node in the graph. The node indexed by the first element of each row is mapped to the second element in each row

Returns

np.ndarray The permuted graph.

wlalign.alignment.**length_wl_signature**(*k: int*, *l: int*) → int

Returns the length of the WL signature corresponding to the width and depth parameters given as input.

Parameters

- **k** –

  int Width of the breadth-first search.

- **l** –

  int Depth of the breadth-first search.

Returns

int Length of the WL align signature.

Raises

- **ValueError** – if k <= 0.
- **ValueError** – if l <= 0.

wlalign.alignment.**permutation_from_alignment**(*alignment: numpy.ndarray*) → numpy.ndarray

Transform a matching into a permutation matrix.

Parameters **alignment** –

np.ndarray n-by-2 array with one row per node in the graph. The node indexed by the first element of each row is mapped to the second element in each row

Returns

np.ndarray Permutation matrix corresponding to the alignment

wlalign.alignment.**signature_wl**(*g: numpy.ndarray*, *k: int*, *l: int*, *node: int*, *volumes: Optional[list] = None*) → numpy.ndarray

Compute the WL-align signature of a node of a graph from its adjacency matrix.

Parameters

- **g** –

  np.ndarray Adjacency matrix of the graph.

- **k** –

  int Width parameter of the breadth-first search.

- **l** –

  int Depth parameter of the breadth-first search

- **node** –

  int Index of the node of which the signature must be computed.

- **volumes** –

  Optional[list] List containing the volume of each node. If not passed, it's computed on the fly from the adjacency matrix.

Returns

np.ndarray Signature of the node.

References

Matteo Frigo, Emilio Cruciani, David Coudert, Rachid Deriche, Emanuele Natale, Samuel Deslauriers-Gauthier; Network alignment and similarity reveal atlas-based topological differences in structural connectomes. Network Neuroscience 2021; doi: https://doi.org/10.1162/netn_a_00199

wlalign.alignment.**wl_align**(*g1: numpy.ndarray*, *g2: numpy.ndarray*, *k: int*, *l: int*) → numpy.ndarray
Compute the WL alignment between two graphs as in Frigo et al., 2021.

Parameters

- **g1** –

  np.ndarray Adjacency matrix of the first graph to align.

- **g2** –

  np.ndarray Adjacency matrix of the second graph to align.

- **k** –

  int Width parameter of the breadth-first search.

- **l** –

  int Depth parameter of the breadth-first search

Returns

np.ndarray Matrix with two columns and one row per node. The first element of each row is the index of the node in the first graph that is aligned with the node in the second graph indexed by the second element of the row.

References

Matteo Frigo, Emilio Cruciani, David Coudert, Rachid Deriche, Emanuele Natale, Samuel Deslauriers-Gauthier; Network alignment and similarity reveal atlas-based topological differences in structural connectomes. Network Neuroscience 2021; doi: https://doi.org/10.1162/netn_a_00199

3.4 Similarity module

wlalign.similarity.**graph_jaccard_index**(*g1*, *g2*)
Compute the Graph Jaccard Index (GJI) between two graphs.

Parameters

- **g1** –

  np.ndarray Adjacency matrix of the first graph.

- **g2** –

  np.ndarray Adjacency matrix of the second graph.

Returns

float Value of the GJI between the two graphs. The value will be in the [0, 1] range.

References

Matteo Frigo, Emilio Cruciani, David Coudert, Rachid Deriche, Emanuele Natale, Samuel Deslauriers-Gauthier; Network alignment and similarity reveal atlas-based topological differences in structural connectomes. Network Neuroscience 2021; doi: https://doi.org/10.1162/netn_a_00199

3.5 Utils module

wlalign.utils.**check_can_write_file**(*fpath: str*, *force: bool = False*) → None
Check if a file can be written. The function checks if the file already exists, the user has the permission to write it, overwriting can be forced and, if the file does not exist, if the parent directory exists and is writable.

Parameters

- **fpath** –

  str Path of the file to be checked.

- **force** –

  bool True if the file can be overwritten, False otherwise.

Raises

- **FileExistsError** – if the file exists and can not be overwritten.
- **PermissionError** – if the file esists and the user does not have the permission to write it.
- **PermissionError** – if the file does not exist, the parent directory exists and the user does not have the permission to write a file in it.
- **FileNotFoundError** – if file does not exist and the parent directory does not exist.

wlalign.utils.**check_compatible_adj**(*g1: numpy.ndarray*, *g2: numpy.ndarray*)
Check if two graphs have the same number of nodes.

Parameters

- **g1** –

  np.ndarray First graph to compare.

- **g2** –

  np.ndarray Second graph to compare.

Raises **ValueError** – if the two adjacency matrices do not have the same number of rows and columns.

wlalign.utils.**check_is_adj**(*g: numpy.ndarray*)
Check if a matrix is an adjacency matrix

Parameters **g** –

np.ndarray Matrix to be checked.

Raises

- **ValueError** – if the matrix is not 2-dimensional.
- **ValueError** – if the matrix is not square.

wlalign.utils.**check_is_alignment**(*alignment: numpy.ndarray*, *n: Optional[int] = None*)
Check if a numpy array is a valid alignment.

Parameters

- **alignment** –

  np.ndarray Alignment to be checked. Must be n-by-2 numpy array.

- **n** –

  Optional[int] Number of nodes expected to be found in the alignment.

Raises

- **ValueError** – if the alignment does not have 2 dimensions.
- **ValueError** – if the alignment does not have 2 columns.
- **ValueError** – if the alignment does not have n rows.
- **ValueError** – if the alignment is not a 1-1 correspondence.

wlalign.utils.**load_network**(*fpath: str*, *delimiter: str = ' '*, *skip: str = '#'*) → numpy.ndarray

Load a network from a text file with the adjacency matrix

Parameters

- **fpath** –

  str Path of the file where the adjacency matrix is saved.

- **delimiter** –

  str Character that separates two consecutive entries in a row. Default: whitespace.

- **skip** –

  str Lines that start with this character will be skipped.

Returns

np.ndarray The adjacency matrix.

wlalign.utils.**symmetrize_adj**(*g: numpy.ndarray*) → numpy.ndarray

Transform a graph from directed to undirected by summing the weights in the two directions of each edge.

Parameters **g** –

np.ndarray Adjacency matrix of the graph to symmetrize.

Returns

np.ndarray Adjacency matrix of the symmetrized graph.

3.6 How to cite WL-align

If you use WL-align in your research, please cite the following article.

Matteo Frigo, Emilio Cruciani, David Coudert, Rachid Deriche, Emanuele Natale, Samuel Deslauriers-Gauthier; Network alignment and similarity reveal atlas-based topological differences in structural connectomes. Network Neuroscience 2021; DOI: 10.1162/netn_a_00199

```
@article{wlalign,
      author = {Frigo, Matteo and Cruciani, Emilio and Coudert, David and
                Deriche, Rachid and Natale, Emanuele and
                Deslauriers-Gauthier, Samuel},
```

(continues on next page)

(continued from previous page)

```
    title = {Network alignment and similarity
             reveal atlas-based topological differences in structural
             connectomes},
    journal = {Network Neuroscience},
    url = {https://hal.archives-ouvertes.fr/hal-03116143},
    doi = {10.1162/netn_a_00199},
    year = {2021}
}
```

3.7 List of Contributors

WL-align was conceived in the COATI and ATHENA Project Teams at Inria Sophia Antipolis - Méditerranée. The Python package was developed by:

- Matteo Frigo , ATHENA Project Team, Inria Sophia Antipolis - Méditerranée.
- Emilio Cruciani , COATI Project Team, Inria Sophia Antipolis - Méditerranée.

3.8 License

3.9 Funding

The development of WL-align was funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (ERC Advanced Grant agreement No 694665: CoBCoM - Computational Brain Connectivity Mapping).


erc
European Research Council
Established by the European Commission

CHAPTER

FOUR

FUNDING

The development of WL-align was funded by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (ERC Advanced Grant agreement No 694665: CoBCoM - Computational Brain Connectivity Mapping).

PYTHON MODULE INDEX

W

INDEX