Journal of Machine Learning Research 21 (2020) 1-6 Submitted 4/20; Published 9/20

Scikit-network: Graph Analysis in Python

Thomas Bonald THOMAS.BONALD@TELECOM-PARIS.FR
Nathan de Lara NATHAN.DELARA@TELECOM-PARIS.FR
Quentin Lutz* QUENTIN.LUTZ@TELECOM-PARIS.FR
Télécom Paris
Institut Polytechnique de Paris
91120 Palaiseau - France
Bertrand Charpentier BERTRAND.CHARPENTIER@IN.TUM.DE
Technical University of Munich
D-80333 Munich - Germany

Editor: Andreas Mueller

Abstract

Scikit-network is a Python package inspired by scikit-learn for the analysis of large graphs. Graphs are represented by their adjacency matrix in the sparse CSR format of SciPy. The package provides state-of-the-art algorithms for ranking, clustering, classifying, embedding and visualizing the nodes of a graph. High performance is achieved through a mix of fast matrix-vector products (using SciPy), compiled code (using Cython) and parallel processing. The package is distributed under the BSD license, with dependencies limited to NumPy and SciPy. It is compatible with Python 3.6 and newer. Source code, documentation and installation instructions are available online[1].

Keywords: graph analysis, sparse matrices, python, cython, scipy

1. Introduction

Scikit-learn (Pedregosa et al., 2011) is a machine learning package based on the popular Python language. It is well-established in today's machine learning community thanks to its versatility, performance and ease of use, making it suitable for both researchers, data scientists and data engineers. Its main assets are the variety of algorithms, the performance of their implementation and their common API.

Scikit-network is a Python package inspired by scikit-learn for graph analysis. The sparse nature of real graphs, with up to millions of nodes, prevents their representation as dense matrices and rules out most algorithms of scikit-learn. *Scikit-network* takes as input a sparse matrix in the CSR format of SciPy and provides state-of-the-art algorithms for ranking, clustering, classifying, embedding and visualizing the nodes of a graph.

The design objectives of *scikit-network* are the same as those having made scikit-learn a success: versatility, performance and ease of use. The result is a Python-native package, like NetworkX (Hagberg et al., 2008), that achieves the state-of-the-art performance of iGraph (Csardi and Nepusz, 2006) and graph-tool (Peixoto, 2014) (see the benchmark in section 5). *Scikit-*

*This author is also affiliated with Nokia Bell Labs France.

[1] See `https://scikit-network.readthedocs.io/en/latest/`.

network uses the same API as Scikit-learn, with algorithms available as classes with the same methods (e.g., `fit`). It is distributed with the BSD license, with dependencies limited to NumPy (Walt et al., 2011) and SciPy (Virtanen et al., 2020).

2. Software Features

The package is organized in modules with consistent API, covering various tasks:

- **Data.** Module for loading graphs from distant repositories, including Konect (Kunegis, 2013), parsing `tsv` files into graphs, and generating graphs from standard models, like the stochastic block model (Airoldi et al., 2008).
- **Clustering.** Module for clustering graphs, including a soft version that returns a node-cluster membership matrix.
- **Hierarchy.** Module for the hierarchical clustering of graphs, returning dendrograms in the standard format of SciPy. The module also provides various post-processing algorithms for cutting and compressing dendrograms.
- **Embedding.** Module for embedding graphs in a space of low dimension. This includes spectral embedding and standard dimension reduction techniques like SVD and GSVD, with key features like regularization.
- **Ranking.** Module for ranking the nodes of the graph by order of importance. This includes PageRank (Page et al., 1999) and various centrality scores.
- **Classification.** Module for classifying the nodes of the graph based on the labels of a few nodes (semi-supervised learning).
- **Path.** Module relying on SciPy for the shortest path problems.
- **Topology.** Module for exploring the structure of the graph: graph traversals, connected components, etc.
- **Visualization.** Module for visualizing graphs and dendrograms in SVG (Scalable Vector Graphics) format. Examples are displayed in Figure 1.

These modules are only partially covered by existing graph softwares (see Table 1). Another interesting feature of *scikit-network* is its ability to work directly on bipartite graphs, represented by their biadjacency matrix.

3. Project Assets

Code quality and availability. Code quality is assessed by standard code coverage metrics. Today's coverage is at 98% for the whole package. Requirements are also kept up to date thanks to the PyUp tool. *Scikit-network* relies on TravisCI for continuous integration and cibuildwheel and manylinux for deploying on common platforms. OSX, Windows 32 or 64-bit and most Linux distributions (McGibbon and Smith, 2016) are supported for Python versions 3.6 and newer.

Modules	scikit-network	NetworkX	iGraph	graph-tool
Data	✓	✓	✗	✓
Clustering	✓	✓	✓	✗
Hierarchy	✓	✗	✓	✓
Embedding	✓	✓	✗	✓
Ranking	✓	✓	✓	✓
Classification	✓	✓	✗	✗
Path	✓	✓	✓	✓
Topology	✓	✓	✓	✓
Visualization	✓	✓	✓	✓

Table 1: Overview of graph software features. ✓: Available. ✓: Partially available or slow implementation. ✗: Not available.

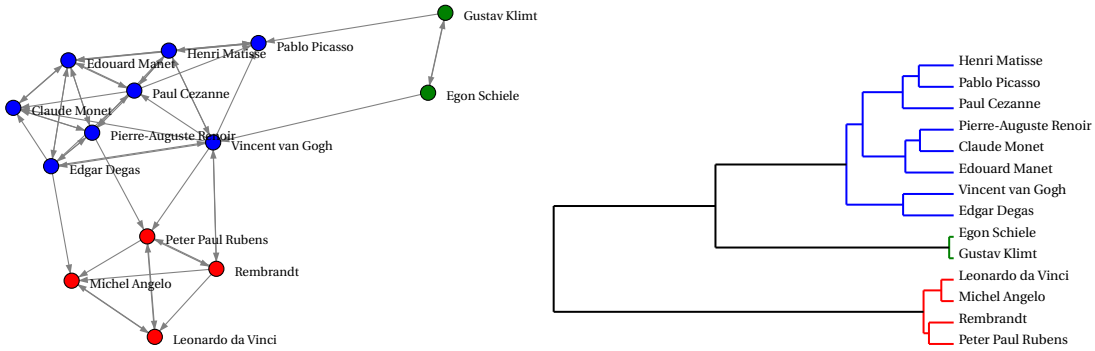

Figure 1: Visualization of a graph and a dendrogram as SVG images.

Open-source software. The package is hosted on GitHub[2] and part of SciPy kits aimed at creating open-source scientific software. Its BSD license enables maximum interoperability with other software. Guidelines for contributing are described in the package's documentation[3] and guidance is provided by the GitHub-hosted Wiki.

Documentation. *Scikit-network* is provided with a complete documentation[3]. The API reference presents the syntax while the tutorials present applications on real graphs. Algorithms are documented with relevant formulas, specifications, examples and references, when relevant.

Code readability. The source code follows the stringent PEP8 guidelines. Explicit variable naming and type hints make the code easy to read. The number of object types is kept to a minimum.

Data collection. The package offers multiple ways to fetch data. Some small graphs are embedded in the package itself for testing or teaching purposes. Part of the API makes it possible to fetch data from selected graph databases easily. Parsers are also present to enable users to import their own data and save it in a convenient format for later reuse.

[2] See `https://github.com/sknetwork-team/scikit-network`.
[3] See `https://scikit-network.readthedocs.io/en/latest/`.

	scikit-network	NetworkX	iGraph	graph-tool
Louvain	771	✗	1,978	✗
PageRank	48		236	45
HITS	109		80	144
Spectral	534		✗	✗

Table 2: Execution times (in seconds). ✗: Not available. : Memory overflow.

	scikit-network	NetworkX	iGraph	graph-tool
RAM usage	1,222		17,765	10,366

Table 3: Memory usage (in MB). : Memory overflow.

4. Resources

Scikit-network relies on a very limited number of external dependencies for ease of installation and maintenance. Only SciPy and NumPy are required on the user side.

SciPy. Many elements from SciPy are used for both high performance and simple code. The sparse matrix representations allow for efficient manipulations of large graphs while the linear algebra solvers are used in many algorithms. Scikit-network also relies on the *LinearOperator* class for efficient implementation of certain algorithms.

NumPy. NumPy arrays are used through SciPy's sparse matrices for memory-efficient computations. NumPy is used throughout the package for the manipulation of arrays. Some inputs and most of the outputs are given in the NumPy array format.

Cython. In order to speed up execution times, Cython (Behnel et al., 2011) generates C++ files automatically using a Python-like syntax. Thanks to the Python wheel system, no compilation is required from the user on most platforms. Note that Cython has a built-in module for parallel computing on which *scikit-network* relies for some algorithms. Otherwise, it uses Python's native multiprocessing.

5. Performance

To show the performance of *scikit-network*, we compare the implementation of some representative algorithms with those of the graph softwares of Table 1: the Louvain clustering algorithm (Blondel et al., 2008), PageRank (Page et al., 1999), HITS (Kleinberg, 1999) and the spectral embedding (Belkin and Niyogi, 2003). For PageRank, the number of iterations is set to 100 when possible (that is for all packages except iGraph). For Spectral, the dimension of the embedding space is set to 16.

Table 2 gives the running times of these algorithms on the *Orkut* graph of Konect (Kunegis, 2013). The graph has $3,072,441$ nodes and $117,184,899$ edges. The computer has a Debian 10 OS and is equipped with an AMD Ryzen Threadripper 1950X 16-Core Processor and 32 GB of RAM. As we can see, *scikit-network* is highly competitive.

We also give in Table 3 the memory usage of each package when loading the graph. Thanks to the CSR format, *scikit-network* has a minimal footprint.

Acknowledgments

This work is supported by Nokia Bell Labs (CIFRE convention 2018/1648).

References

Edoardo M Airoldi, David M Blei, Stephen E Fienberg, and Eric P Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9(Sep):1981–2014, 2008.

Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, and Kurt Smith. Cython: The best of both worlds. *Computing in Science & Engineering*, 13(2):31–39, 2011.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, October 2008. ISSN 1742-5468. doi: 10.1088/1742-5468/2008/10/P10008. URL https://doi.org/10.1088%2F1742-5468%2F2008%2F10%2Fp10008.

Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL http://igraph.org.

Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

Jérôme Kunegis. KONECT: The Koblenz Network Collection. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13 Companion, pages 1343–1350, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2038-2. doi: 10.1145/2487788.2488173. URL http://doi.acm.org/10.1145/2487788.2488173. event-place: Rio de Janeiro, Brazil.

Robert T. McGibbon and Nathaniel J. Smith. A platform tag for portable linux built distributions. PEP 513, -, 2016. URL https://www.python.org/dev/peps/pep-0513/.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

Tiago P. Peixoto. The graph-tool python library. *figshare*, 2014. doi: 10.6084/m9.figshare.1164194. URL http://figshare.com/articles/graph_tool/1164194.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: https://doi.org/10.1038/s41592-019-0686-2.

Stéfan J. van der Walt, S Chris Colbert, and Gaël Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.