The Journal of Open Source Software

scikit-hubness: Hubness Reduction and Approximate Neighbor Search

Roman Feldbauer[1], **Thomas Rattei**[1], **and Arthur Flexer**[2]

1 Division of Computational Systems Biology, Department of Microbiology and Ecosystem Science, University of Vienna, Althanstraße 14, 1090 Vienna, Austria **2** Austrian Research Institute for Artificial Intelligence (OFAI), Freyung 6/6/7, 1010 Vienna, Austria

DOI: 10.21105/joss.01957

Software

- Review
- Repository
- Archive

Editor: Yuan Tang

Reviewers:

- @ryEllison
- @aozorahime

Submitted: 10 December 2019
Published: 14 January 2020

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC-BY).

Summary

`scikit-hubness` is a Python package for efficient nearest neighbor search in high-dimensional spaces. Hubness is an aspect of the *curse of dimensionality* in nearest neighbor graphs. Specifically, it describes the increasing occurrence of *hubs* and *antihubs* with growing data dimensionality: Hubs are objects, that appear unexpectedly often among the nearest neighbors of others objects, while antihubs are never retrieved as neighbors. As a consequence, hubs may propagate their information (for example, class labels) too widely within the neighbor graph, while information from antihubs is depleted. These semantically distorted graphs can reduce learning performance in various tasks, such as classification (Radovanović, Nanopoulos, & Ivanović, 2010), clustering (Schnitzer & Flexer, 2015), or visualization (Flexer, 2015). Hubness is known to affect a variety of applied learning systems (Angiulli, 2018), causing — for instance — overrepresentation of certain songs in music recommendations (Flexer & Stevens, 2018), or improper transport mode detection (Feldbauer, Leodolter, Plant, & Flexer, 2018).

Multiple hubness reduction algorithms have been developed to mitigate these effects (Flexer & Schnitzer, 2013; Hara, Suzuki, Kobayashi, Fukumizu, & Radovanovic, 2016; Schnitzer, Flexer, Schedl, & Widmer, 2012). We compared these algorithms exhaustively in a recent survey (Feldbauer & Flexer, 2019), and developed approximate hubness reduction methods with linear time and memory complexity (Feldbauer et al., 2018).

Currently, there is a lack of fully-featured, up-to-date, user-friendly software dealing with hubness. Available packages miss critical features and have not been updated in years ("Hub-Miner," 2015), or are not particularly user-friendly ("Hub-Toolbox," 2019). In this paper we describe `scikit-hubness`, which provides powerful, readily available, and easy-to-use hubness-related methods:

- hubness analysis ("Is my data affected by hubness?"): Assess hubness with several measures, including k-occurrence skewness (Radovanović et al., 2010), and Robin-Hood index (Feldbauer et al., 2018).
- hubness reduction ("How can we improve neighbor retrieval in high dimensions?"): Mutual proximity, local scaling, and DisSimLocal are currently supported, as they performed best in our survey. Exact methods as well as their approximations are available.
- approximate neighbor search ("Does it work for large data sets?"): Several methods are currently available, including locality-sensitive hashing (Aumüller, Christiani, Pagh, & Vesterli, 2019) and hierarchical navigable small-world graphs (Malkov & Yashunin, 2018).

`scikit-hubness` builds upon the SciPy stack (Virtanen, Gommers, Oliphant, Haberland, & others, 2019) and is integrated into the `scikit-learn` environment (Pedregosa, Varoquaux, Gramfort, Michel, & others, 2011), enabling rapid adoption by Python-based machine learning researchers and practitioners. Convenient interfaces to hubness-reduced neighbors-based learning are available in the `skhubness.neighbors` subpackage. It acts as a drop-in replacement for `sklearn.neighbors`, featuring all its functionality, and adding support for hubness reduction, where applicable. This includes, for example, the supervised `KNeighborsClassifier` and `RadiusNeighborsRegressor`, `NearestNeighbors` for unsupervised learning, and the general `kneighbors_graph`.

`scikit-hubness` is developed using several quality assessment tools and principles, such as PEP8 compliance, unit tests with high code coverage, continuous integration on all major platforms (Linux and MacOS [1], Windows [2]), and additional checks by LGTM [3]. The source code is available at GitHub [4] under the BSD 3-clause license. The online documentation is available at Read the Docs [5]. Install from the Python package index with `$ pip install scikit-hubness`.

[1] https://travis-ci.com/VarIr/scikit-hubness/

[2] https://ci.appveyor.com/project/VarIr/scikit-hubness

[3] https://lgtm.com/projects/g/VarIr/scikit-hubness/

[4] https://github.com/VarIr/scikit-hubness

[5] https://scikit-hubness.readthedocs.io/

Outlook

Future plans include adaption to significant changes of `sklearn.neighbors` introduced in version 0.22 in December 2019: The `KNeighborsTransformer` and `RadiusNeighborsTransformer` transform data into sparse neighbor graphs, which can subsequently be used as input to other estimators. Hubness reduction and approximate search can then be implemented as `Transformers`. This provides the means to turn `skhubness.neighbors` from a drop-in replacement of `sklearn.neighbors` into a scikit-learn plugin, which will (1) accelerate development, (2) simplify addition of new hubness reduction and approximate search methods, and (3) facilitate more flexible usage.

Acknowledgements

We thank Silvan David Peter for testing the software. This research is supported by the Austrian Science Fund (FWF): P27703 and P31988

References

Angiulli, F. (2018). On the behavior of intrinsically high-dimensional spaces: Distances, direct and reverse nearest neighbors, and hubness. *J. Mach. Learn. Res.*, *18*, 170:1–170:60.

Aumüller, M., Christiani, T., Pagh, R., & Vesterli, M. (2019). PUFFINN: parameterless and universally fast finding of nearest neighbors. In *27th annual european symposium on algorithms, ESA* (Vol. 144, pp. 10:1–10:16). doi:10.4230/LIPIcs.ESA.2019.10

Feldbauer, R., & Flexer, A. (2019). A comprehensive empirical comparison of hubness reduction in high-dimensional spaces. *Knowledge and Information Systems*, *59*(1), 137. doi:10.1007/s10115-018-1205-y

Feldbauer, R., Leodolter, M., Plant, C., & Flexer, A. (2018). Fast approximate hubness reduction for large high-dimensional data. In *Proc. IEEE Int. Conf. Big Knowledge* (pp. 358–367). doi:10.1109/ICBK.2018.00055

Flexer, A. (2015). Improving visualization of high-dimensional music similarity spaces. In *Proceedings of the 16th international society for music information retrieval conference, ISMIR 2015* (pp. 547–553).

Flexer, A., & Schnitzer, D. (2013). Can shared nearest neighbors reduce hubness in high-dimensional spaces? In *Proc. IEEE 13th Int. Conf. Data Mining workshops* (pp. 460–467). doi:10.1109/ICDMW.2013.101

Flexer, A., & Stevens, J. (2018). Mutual proximity graphs for improved reachability in music recommendation. *Journal of New Music Research*, *47*(1), 17–28. doi:10.1080/09298215.2017.1354891

Hara, K., Suzuki, I., Kobayashi, K., Fukumizu, K., & Radovanovic, M. (2016). Flattening the density gradient for eliminating spatial centrality to reduce hubness. In *AAAI conference on artificial intelligence*.

HubMiner. (2015). Retrieved from http://ailab.ijs.si/tools/hub-miner/

Hub-Toolbox. (2019). Retrieved from https://github.com/OFAI/hub-toolbox-python3/

Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1. doi:10.1109/TPAMI.2018.2889473

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., & others. (2011). Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, *12*, 2825–2830.

Radovanović, M., Nanopoulos, A., & Ivanović, M. (2010). Hubs in space: Popular nearest neighbors in high-dimensional data. *J. Mach. Learn. Res.*, *11*, 2487–2531.

Schnitzer, D., & Flexer, A. (2015). The unbalancing effect of hubs on k-medoids clustering in high-dimensional spaces. In *Proc. Int. Joint Conf. Neural Networks (IJCNN)* (pp. 1–8). doi:10.1109/IJCNN.2015.7280303

Schnitzer, D., Flexer, A., Schedl, M., & Widmer, G. (2012). Local and global scaling reduce hubs in space. *J. Mach. Learn. Res.*, *13*(1), 2871–2902.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., & others. (2019). SciPy 1.0–fundamental algorithms for scientific computing in Python. *arXiv preprint*. Retrieved from http://arxiv.org/abs/1907.10121v1