`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*
{*roman.feldbauer,thomas.rattei*}*@univie.ac.at, arthur.flexer@ofai.at*

Abstract

This paper introduces `scikit-hubness`, a Python package for efficient nearest neighbor search in high-dimensional spaces. Hubness is an aspect of the *curse of dimensionality*, and is known to impair various learning tasks, including classification, clustering, and visualization. `scikit-hubness` provides algorithms for hubness analysis ("Is my data affected by hubness?"), hubness reduction ("How can we improve neighbor retrieval in high dimensions?"), and approximate neighbor search ("Does it work for large data sets?"). It is integrated into the `scikit-learn` environment, enabling rapid adoption by Python-based machine learning researchers and practitioners. Users will find all functionality of the `scikit-learn neighbors` package, plus additional support for transparent hubness reduction and approximate nearest neighbor search. `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, MacOS, Windows), and additional checks by LGTM. The source code is available at `https://github.com/VarIr/scikit-hubness` under the BSD 3-clause license. Install from the Python package index with `$ pip install scikit-hubness`.

Key words hubness, nearest neighbors, curse of dimensionality, scikit-learn, Python

1 Introduction

Hubness is a phenomenon in nearest neighbor graphs (Radovanović et al., 2010), that is often detrimental to data mining and learning tasks building upon them. It is an aspect of the *curse of dimensionality*, which describes challenges arising when data are embedded in high dimensions. Specifically, hubness describes the increasing occurrence of *hubs* and *antihubs* with increasing data dimensionality: Hubs are objects, that appear uncommonly 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 et al., 2010), clustering (Schnitzer and Flexer, 2015),

or visualization (Flexer, 2015). Hubness is known to affect a variety of applied learning systems,causing—for instance—odd music recommendations (Schnitzer et al., 2012), or improper transport mode detection (Feldbauer et al., 2018).

Multiple hubness reduction algorithms have been developed to mitigate these effects (Schnitzer et al., 2012; Flexer and Schnitzer, 2013; Hara et al., 2015, 2016). In a recent survey, we compared these algorithms exhaustively (Feldbauer and Flexer, 2019), and developed approximate hubness reduction methods with linear time and memory complexity for the most successful algorithms (Feldbauer et al., 2018).

In this paper we describe `scikit-hubness`, which provides readily available, easy-to-use hubness analysis and reduction for both machine learning researchers and practitioners.

2 Software Development and Architecture

`scikit-hubness` is a Python package built upon the SciPy stack (Virtanen et al., 2019) and scikit-learn (Pedregosa et al., 2011) with cross-platform support for Linux, MacOS, and Windows. It consists of three sub-packages, that are detailed in the sections below. Code style and API design is based on the guidelines of `scikit-learn`, with PEP 8 compliance, NumPy documentation format, and additional criteria, such as those tested by LGTM.[1] Code development is aided by continuous integration on all three platforms, with an extensive set of tests covering nearly the complete code base. Source code, issue tracker, quickstart, and additional links are available at GitHub.[2] The online documentation is available at Read the Docs.[3]

2.1 `analysis`: Hubness Analysis

The `skhubness.analysis` sub-package allows to determine, whether data is affected by hubness. This is often the first step in a user's workflow, allowing for quick assessment of potential improvements due to hubness reduction. The package provides several hubness measures, including long-proven ones, such as the k-occurrence skewness (Radovanović et al., 2010), as well as recently proposed measures, such as the Robin-Hood index (Feldbauer et al., 2018).

2.2 `reduction`: Hubness Reduction Algorithms

The `skhubness.reduction` sub-package provides hubness reduction algorithms. At the time of writing, the most successful methods (Feldbauer and Flexer, 2019) are available: mutual proximity, local scaling (Schnitzer et al., 2012), and $\text{DisSim}^{\text{Local}}$ (Hara et al., 2016). We implement both the exact methods (quadratic complexity), and their approximations (linear complexity, Feldbauer et al., 2018). Neighbor graphs can be hubness-reduced directly with classes from this package, which may be especially useful for hubness-related research.

[1] Automated code analysis is performed by `https://lgtm.com/projects/g/VarIr/scikit-hubness/`.

[2] Source code, issue tracker, and additional links etc. can be found at `https://github.com/VarIr/scikit-hubness/`.

[3] Online documentation is available at `https://scikit-hubness.readthedocs.io/`.

```python
from skhubness.data import\
    load_dexter
X, y = load_dexter()

from sklearn.neighbors import\
    KNeighborsClassifier
knn_vanilla = KNeighborsClassifier(
    n_neighbors=5, metric="cosine",
)

from sklearn.model_selection import\
    cross_val_score
acc_vanilla = cross_val_score(
    knn_vanilla, X, y, cv=5)

# Accuracy (vanilla kNN):
print(f"{acc_vanilla.mean():.3f}")
# 0.793
```

```python
from skhubness.data import\
    load_dexter
X, y = load_dexter()

from skhubness.neighbors import\
    KNeighborsClassifier
knn_mp = KNeighborsClassifier(
    n_neighbors=5, metric="cosine",
    hubness="mutual_proximity")

from sklearn.model_selection import\
    cross_val_score
acc_mp = cross_val_score(
    knn_mp, X, y, cv=5)

# Accuracy (hubness-reduced kNN)
print(f"{acc_mp.mean():.3f}")
# 0.893
```

Example 1: Quickstart example. Left: kNN classification in `scikit-learn`. Right: Hubness reduction is enabled in `scikit-hubness` by providing one additional parameter.

2.3 `neighbors`: Neighbors-Based Learning

Practitioners find convenient interfaces to hubness-reduced neighbors-based learning in the `skhubness.neighbors` sub-package. It features all functionality from `sklearn.neighbors` and adds support for hubness reduction, where applicable. This includes, for example, the supervised `KNeighborsClassifier` and `RadiusNeighborsRegressor`, `NearestNeighbors` for unsupervised learning, and the general `kneighbors_graph`.

Time and memory requirements of exact nearest neighbor algorithms scale quadratically with the number of samples. In order to support very large data sets, scikit-hubness provides approximate neighbor search with linear complexity. Several methods are currently available, including locality-sensitive hashing and hierarchical navigable small-world graphs(Aumüller et al., 2019; Malkov and Yashunin, 2016). Note, that approximate search can be employed in conjunction with or independently from hubness reduction. Therefore, `scikit-hubness` provides powerful neighbor search in large-scale data, hubness-affected or not.

3 Installation and Basic Example

Assuming an existing Python environment, `scikit-hubness` can be installed from the Python package index via `$ pip install scikit-hubness` at the command line.

Example 1 shows a quickstart example of text classification. Adapting `sklearn`-based nearest neighbor pipelines to use hubness reduction requires only minor modifications (see highlighted code). Given the vast user base of scikit-learn, hubness reduction is now available to a large number of machine learning researchers and practitioners with very little transition effort.

Tool	Platform	Last update	Mutual proximity	Local scaling	DisSim Local	Approximate hubness reduction	scikit-learn compatible
HubMiner[4]	Java	2015	✓	✓	✗	✗	(N/A)
Hub-Toolbox[5]	Matlab	2016	✓	✓	✗	✗	(N/A)
Hub-Toolbox[6]	Python	2019	✓	✓	✓	✓	(partial)
scikit-hubness	Python	2019	✓	✓	✓	✓	✓

Table 1: Comparison of hubness reduction software packages

4 Comparison to Similar Software

Several tools for hubness reduction have previously been developed. Table 1 provides an overview of selected features. Within the last three years, only the Hub-Toolbox for Python and scikit-hubness have been actively developed. Both tools originate from our research groups. The Hub-Toolbox was released primarily to enable reproducibility for hubness research performed in our labs. It features only partial compatibility with scikit-learn, and is currently lacking a consistent API. Scikit-hubness succeeds the Hub-Toolbox, and puts a strong focus on usability. It has been rewritten from scratch in order to provide a more user-friendly, more easily extensible, fully scikit-learn compatible machine learning package.

5 Conclusion and Outlook

Ever since its original release, scikit-learn has become one of the most popular machine learning frameworks with over 65 000 dependent repositories on GitHub at the time of writing (August 2019). Scikit-hubness integrates seamlessly with scikit-learn, and introduces effective hubness reduction and approximate neighbor search into this ecosystem. We are confident to provide the most user-friendly hubness analysis and reduction software package released so far.

Future plans include adaption to significant changes of `sklearn.neighbors` likely to be introduced in the next major release: The `KNeighborsTransformer` and `RadiusNeighborsTransformer`[7] 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

[4]HubMiner is available from `http://ailab.ijs.si/tools/hub-miner/`.
[5]Hub-Toolbox for MATLAB is available from `https://github.com/OFAI/hub-toolbox-matlab`.
[6]Hub-Toolbox for Python is available from `https://github.com/OFAI/hub-toolbox-python3`.
[7]Upcoming scikit-learn changes: `https://github.com/scikit-learn/scikit-learn/pull/10482`

References

Martin Aumüller, Tobias Christiani, Rasmus Pagh, and Michael Vesterli. PUFFINN: parameterless and universally fast finding of nearest neighbors. In *Annual European Symposium on Algorithms, ESA*, volume 144, pages 10:1–10:16, 2019.

Roman Feldbauer and Arthur Flexer. A comprehensive empirical comparison of hubness reduction in high-dimensional spaces. *Knowledge and Information Systems*, 59(1):137, April 2019.

Roman Feldbauer, Maximilian Leodolter, Claudia Plant, and Arthur Flexer. Fast approximate hubness reduction for large high-dimensional data. In *IEEE International Conference on Big Knowledge (ICBK)*, pages 358–367, November 2018.

Arthur Flexer. Improving visualization of high-dimensional music similarity spaces. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 547–553, 2015.

Arthur Flexer and Dominik Schnitzer. Can shared nearest neighbors reduce hubness in high-dimensional spaces? In *IEEE International Conference on Data Mining (ICDM) Workshops*, pages 460–467, December 2013.

Kazuo Hara, Ikumi Suzuki, Masashi Shimbo, Kei Kobayashi, Kenji Fukumizu, and Miloš Radovanović. Localized centering: Reducing hubness in large-sample data. In *Conference on Artificial Intelligence (AAAI)*, 2015.

Kazuo Hara, Ikumi Suzuki, Kei Kobayashi, Kenji Fukumizu, and Milos Radovanovic. Flattening the density gradient for eliminating spatial centrality to reduce hubness. In *Conference on Artificial Intelligence (AAAI)*, 2016.

Yury Malkov and Dmitry Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *arXiv preprint*, 2016. URL `https://arxiv.org/abs/1603.09320`.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, et al. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, November 2011.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Hubs in space: Popular nearest neighbors in high-dimensional data. *J. Mach. Learn. Res.*, 11:2487–2531, December 2010.

Dominik Schnitzer and Arthur Flexer. The unbalancing effect of hubs on k-medoids clustering in high-dimensional spaces. In *International Joint Conference on Neural Networks (IJCNN)*, July 2015.

Dominik Schnitzer, Arthur Flexer, Markus Schedl, and Gerhard Widmer. Local and global scaling reduce hubs in space. *J. Mach. Learn. Res.*, 13(1):2871–2902, October 2012.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, et al. SciPy 1.0–fundamental algorithms for scientific computing in Python. *arXiv preprint*, 2019. URL `http://arxiv.org/abs/1907.10121v1`.