




HiPart: Hierarchical Divisive Clustering Toolbox

Panagiotis Anagnostou^{1*}, Sotiris Tasoulis^{1*}, Vassilis P. Plagianakos^{1*}, and Dimitris Tasoulis^{2*}

¹ Department of Computer Science and Biomedical Informatics, University of Thessaly, Greece ² Signal Ocean SMPC, Greece ¶ Corresponding author * These authors contributed equally.

DOI: [10.21105/joss.05024](https://doi.org/10.21105/joss.05024)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Mehmet Hakan Satman](#) 



Reviewers:

- [@AP6YC](#)
- [@jjerphan](#)

Submitted: 10 November 2022

Published: 11 April 2023

License

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

Summary

This paper presents the HiPart package, an open-source native Python library that provides efficient and interpretable implementations of divisive hierarchical clustering algorithms. HiPart supports interactive visualizations for the manipulation of the execution steps allowing the direct intervention of the clustering outcome. This package is highly suited for Big Data applications as the focus has been given to the computational efficiency of the implemented clustering methodologies. The dependencies used are either Python build-in packages or highly maintained stable external packages. The software is provided under the MIT license. The package's source code and documentation can be found at [GitHub](#).

Statement of need

A highly researched problem by a variety of research communities is the problem of clustering. However, high-dimensional clustering still constitutes a significant challenge, plagued by the *curse of dimensionality* ([Huttenlocher et al., 2020](#)). Hierarchical divisive algorithms developed in the recent years ([David P. Hofmeyr, 2016](#); [David P. Hofmeyr et al., 2019](#); [David P. Hofmeyr & Pavlidis, 2019](#); [Pavlidis et al., 2016](#); [Tasoulis et al., 2010](#)) have shown great potential for the particular case of high dimensional data, incorporating dimensionality reduction iteratively within their algorithmic procedure. Additionally, they seem unique in providing a hierarchical format of the clustering result with low computational cost, in contrast to the commonly used but computationally demanding agglomerative clustering methods.

Although the discovery of a hierarchical format is crucial in many fields, such as bioinformatics ([Luo et al., 2003](#); [Modena et al., 2014](#)), to the best of our knowledge, this package is the first native Python implementation of divisive hierarchical clustering algorithms. We particularly focus on the “Principal Direction Divisive Clustering (PDDP)” algorithm ([Boley, 1998](#)) for its potential to effectively tackle the *curse of dimensionality* and its impeccable time performance ([Tasoulis et al., 2010](#)).

Simultaneously, we provide implementations of a complete set of hierarchical divisive clustering algorithms with a similar basis. These are the dePDDP ([Tasoulis et al., 2010](#)), the iPDDP ([Tasoulis et al., 2010](#)), the kM-PDDP ([Zeimekis & Gallopoulos, 2008](#)), and the bisecting k-Means (BKM) ([Savarese & Boley, 2001](#)). We also provide additional features which are not included in the original developments of the aforementioned methodologies that make them appropriate for the discovery of arbitrary shaped or non-linear separable clusters. In detail, we incorporate kernel Principal Component Analysis (kPCA) ([Scholkopf et al., 1999](#)) and Independent Component Analysis (ICA) ([Hyvärinen & Oja, 2000](#); [Tharwat, 2020](#)) for the iterative dimensionality reduction steps.

As a result, the package provides a fully parameterized set of algorithms that can be applied in a diverse set of applications.

Software Description

The HiPart (Hierarchical Partitioning) package is divided into three major sections:

- Method implementation
- Static Visualization
- Interactive Visualization

Method Implementation

The package employs an object-oriented approach for the implementation of the algorithms, similarly to that of (Bach et al., 2022), while incorporating design similarities with the scikit-learn library (Pedregosa et al., 2011). Meaning, a class instance executes each of the algorithms, and the class instance's attributes are the algorithm's hyper-parameters and results.

For the execution of the algorithms, users need to call either the `predict` or the `fit_predict` method of each algorithm's execution class. The algorithm parameterization can be applied at the constructor of their respective class.

Static Visualization

Two static visualization methods are included. The first one is a 2-Dimensional representation of all the data splits generated by each algorithm during the hierarchical procedure. The goal is to provide an insight to the users regarding each node of the clustering tree and, subsequently, each step of the algorithm's execution.

The second visualization method is a dendrogram that represents the splits of all the divisive algorithms. The dendrogram's figure creation is implemented by the *SciPy* package (Virtanen et al., 2020), and it is fully parameterized as stated in the library.

Interactive Visualization

In the interactive mode, we provide the possibility for stepwise manipulation of the algorithms. Users can choose a particular step (node of the tree) and manipulate the split-point on top of a two-dimensional visualization, instantly altering the clustering result. Each manipulation resets the algorithm's execution from that step onwards, resulting in a restructuring of the sub-tree of the manipulated node.

Development Notes

For the development of the package, we complied with the **PEP8** style standards, and we enforced it with the employment of *flake8* command-line utility. To ensure the code's quality, we implemented tests using the *pytest* module to assert the correctness of the implementations. In addition, platform compatibility has been assured through extensive testing, and the package development in its entirety uses only well-established or native Python packages. The package has been released as open-source software under the MIT License. For more information regarding potential contributions or for the submission of an issue, or a request, the package is hosted as a repository on Github.

Experiments and Comparisons

In this section, we provide clustering results with respect to the execution speed and clustering performance for the provided implementations. For direct comparison, we employ a series of well-established clustering algorithms. These are the k-Means (Likas et al., 2003), the Agglomerative (AGG) (Ackermann et al., 2014) and the OPTICS (Ankerst et al., 1999)

of the scikit-learn (Pedregosa et al., 2011) Python library and the fuzzy c-means (FCM) algorithm (Bezdek et al., 1984) of the fuzzy-c-means (Dias, 2019) Python package. Clustering performance is evaluated using the Normalized Mutual Information (NMI) score (Yang et al., 2016).

Four widely used data sets from the field of bioinformatics are employed along with two popular data sets benchmark data set for text and image clustering, respectively:

- the Deng (Deng et al., 2014),
- the Baron (Baron et al., 2016),
- the TCGA Pan-cancer (Rala, 2023) (Cancer),
- the Chen (Chen et al., 2017),
- the USPS (Hull, 1994),
- the BBC (Greene & Cunningham, 2006),

All experiments took place on a server computer with Linux operating system, kernel version 5.11.0, with an Intel Core i7-10700K CPU @ 3.80GHz and four DDR4 RAM dims of 32GB with 2133MHz frequency. Default parameters were used for the execution of all the algorithms, and the actual number of clusters was provided to algorithms as a parameter when required.

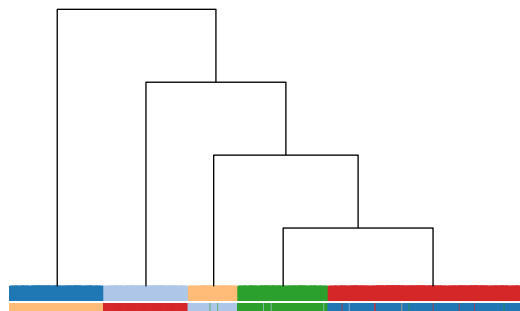


Figure 1: Dendrogram figure for the Cancer data set with the use of the dePDDP algorithm and the dendrogram visualization module of the HiPart library. The line below the tree represents the colour of the original cluster each sample belongs.

In Table 1 we present the mean performance of all methods with respect to execution time (time in secs) and NMI across 100 experiments. We observe that HiPart implementations perform exquisitely in terms of execution time while still being comparable with respect to clustering performance.

Table 1: Clustering results with respect to execution time and clustering performance.

Algorithm	Time (Seconds)	NMI	Time (Seconds)	NMI
Gene Expression Data				
	Deng (135, 12548)		Baron (1886, 14878)	
iPDDP	0.10	0.76	1.16	0.08
dePDDP	0.14	0.70	2.16	0.53
PDDP	0.15	0.54	2.55	0.53
kM-PDDP	0.25	0.61	3.81	0.52
BKM	0.50	0.64	11.51	0.52
k-Means	0.14	0.71	7.52	0.48
AGG	0.04	0.72	14.54	0.51
OPTICS	27.78	0.48	710.99	0.13
FCM	1.37	0.68	163.63	0.45

Algorithm	Time (Seconds)	NMI	Time (Seconds)	NMI
Cancer (801, 20531)			Chen (14437, 23284)	
iPDDP	0.90	0.67	13.18	0.30
dePDDP	1.06	0.93	20.71	0.36
PDDP	1.04	0.74	37.52	0.48
kM-PDDP	1.27	0.86	53.73	0.48
BKM	5.94	0.88	255.85	0.48
k-Means	1.54	0.98	249.72	0.48
AGG	3.09	0.98	1218.68	0.49
OPTICS	266.39	0.34	27089.35	0.00
FCM	5.53	0.53	5710.83	0.26
Benchmark Data				
USPS (4575, 256)			BBC (2225, 21213)	
iPDDP	0.02	0.55	2.02	0.60
dePDDP	0.05	0.65	1.70	0.60
PDDP	0.04	0.60	2.57	0.78
kM-PDDP	0.17	0.50	2.93	0.74
BKM	0.38	0.58	9.92	0.65
k-Means	0.12	0.72	4.68	0.35
AGG	1.15	0.77	23.18	0.64
OPTICS	635.05	0.04	1055.75	0.06
FCM	2.92	0.58	4.73	0.60

Conclusions and Future Work

We present a highly time-efficient clustering package with a suite of tools that give the capability of addressing problems in high-dimensional clustering problems. Also, the developed new visualization tools enhance understanding and identification of the underlying clustering data structure.

We plan to continuously expand the HiPart package in the future through the addition of more hierarchical algorithms and by providing even more options for dimensionality reduction, such as the use of recent projection pursuit methodologies (David P. Hofmeyr, 2016; David P. Hofmeyr et al., 2019; David P. Hofmeyr & Pavlidis, 2019; Pavlidis et al., 2016). Our final aim is to establish the golden standard when considering hierarchical divisive clustering.

Acknowledgments

This project has received funding from the Hellenic Foundation for Research and Innovation (HFRI), under grant agreement No 1901.

References

- Ackermann, M. R., Blömer, J., Kuntze, D., & Sohler, C. (2014). Analysis of agglomerative clustering. *Algorithmica*, 69(1), 184–215.
- Ankerst, M., Breunig, M. M., Kriegel, H.-P., & Sander, J. (1999). OPTICS: Ordering points to identify the clustering structure. *ACM Sigmod Record*, 28(2), 49–60.
- Bach, P., Chernozhukov, V., Kurz, M. S., & Spindler, M. (2022). DoubleML - an object-oriented implementation of double machine learning in Python. *Journal of Machine Learning Research*, 23(53), 1–6. <http://jmlr.org/papers/v23/21-0862.html>

- Baron, M., Veres, A., Wolock, S. L., Faust, A. L., Gaujoux, R., Vetere, A., Ryu, J. H., Wagner, B. K., Shen-Orr, S. S., Klein, A. M., & others. (2016). A single-cell transcriptomic map of the human and mouse pancreas reveals inter- and intra-cell population structure. *Cell Systems*, 3(4), 346–360. <https://doi.org/10.1016/j.cels.2016.08.011>
- Bezdek, J. C., Ehrlich, R., & Full, W. (1984). FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2), 191–203.
- Boley, D. (1998). Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4), 325–344.
- Chen, R., Wu, X., Jiang, L., & Zhang, Y. (2017). Single-cell RNA-seq reveals hypothalamic cell diversity. *Cell Reports*, 18(13), 3227–3241. <https://doi.org/10.1016/j.celrep.2017.03.004>
- Deng, Q., Ramsköld, D., Reinius, B., & Sandberg, R. (2014). Single-cell RNA-seq reveals dynamic, random monoallelic gene expression in mammalian cells. *Science*, 343(6167), 193–196.
- Dias, M. L. D. (2019). *Fuzzy-c-means: An implementation of fuzzy C-means clustering algorithm*. Zenodo. <https://doi.org/10.5281/zenodo.3066222>
- Greene, D., & Cunningham, P. (2006). Practical solutions to the problem of diagonal dominance in kernel document clustering. *Proc. 23rd International Conference on Machine Learning (ICML'06)*, 377–384. <https://doi.org/10.1145/1143844.1143892>
- Hofmeyr, David P. (2016). Clustering by minimum cut hyperplanes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8), 1547–1560. <https://doi.org/10.1109/TPAMI.2016.2609929>
- Hofmeyr, David P., & Pavlidis, N. G. (2019). PPCI: an R Package for Cluster Identification using Projection Pursuit. *The R Journal*, 11(2), 152–170. <https://doi.org/10.32614/RJ-2019-046>
- Hofmeyr, David P., Pavlidis, N. G., & Eckley, I. A. (2019). Minimum spectral connectivity projection pursuit. *Statistics and Computing*, 29(2), 391–414. <https://doi.org/10.1007/s11222-018-9814-6>
- Hull, J. J. (1994). A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5), 550–554. <https://doi.org/10.1109/34.291440>
- Huttenhaler, M., Jentzen, A., Kruse, T., Anh Nguyen, T., & Wursterberger, P. von. (2020). Overcoming the curse of dimensionality in the numerical approximation of semilinear parabolic partial differential equations. *Proceedings of the Royal Society A*, 476(2244), 20190630.
- Hyvärinen, A., & Oja, E. (2000). Independent component analysis: Algorithms and applications. *Neural Networks*, 13(4-5), 411–430.
- Likas, A., Vlassis, N., & Verbeek, J. J. (2003). The global k-means clustering algorithm. *Pattern Recognition*, 36(2), 451–461.
- Luo, F., Tang, K., & Khan, L. (2003). Hierarchical clustering of gene expression data. *Third IEEE Symposium on Bioinformatics and Bioengineering, 2003. Proceedings.*, 328–335.
- Modena, B. D., Tedrow, J. R., Milosevic, J., Bleecker, E. R., Meyers, D. A., Wu, W., Bar-Joseph, Z., Erzurum, S. C., Gaston, B. M., Busse, W. W., & others. (2014). Gene expression in relation to exhaled nitric oxide identifies novel asthma phenotypes with unique biomolecular pathways. *American Journal of Respiratory and Critical Care Medicine*, 190(12), 1363–1372. <https://doi.org/10.1164/rccm.201406-1099OC>
- Pavlidis, N. G., Hofmeyr, D. P., & Tasoulis, S. K. (2016). Minimum density hyperplanes. *Journal of Machine Learning Research*.

- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., & others. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12, 2825–2830.
- Rala, J. R. (2023). *Software-Based Supplementary Materials in MAPEH for Enhanced Learning Resource Management and Delivery System*. Zenodo. <https://doi.org/10.5281/zenodo.7602956>
- Savaresi, S. M., & Boley, D. L. (2001). On the performance of bisecting k-means and PDDP. *Proceedings of the 2001 SIAM International Conference on Data Mining*, 1–14. <https://doi.org/10.1137/1.9781611972719.5>
- Scholkopf, B., Smola, A., & Müller, K.-R. (1999). Kernel principal component analysis. *Advances In Kernel Methods - Support Vector Learning*, 327–352.
- Tasoulis, S. K., Tasoulis, D. K., & Plagianakos, V. P. (2010). Enhancing principal direction divisive clustering. *Pattern Recognition*, 43(10), 3391–3411. <https://doi.org/10.1016/j.patcog.2010.05.025>
- Tharwat, A. (2020). Independent component analysis: An introduction. *Applied Computing and Informatics*.
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17, 261–272. <https://doi.org/10.1038/s41592-019-0686-2>
- Yang, Z., Algesheimer, R., & Tessone, C. J. (2016). A comparative analysis of community detection algorithms on artificial networks. *Scientific Reports*, 6(1), 1–18. <https://doi.org/10.1038/srep30750>
- Zeimpekis, D., & Gallopoulos, E. (2008). Principal direction divisive partitioning with kernels and k-means steering. In *Survey of text mining II* (pp. 45–64). Springer. https://doi.org/10.1007/978-1-84800-046-9_3