The Journal of Open Source Software

cblearn: Comparison-based Machine Learning in Python

David-Elias Künstle[1,2¶] **and Ulrike von Luxburg**[1,2]

1 University of Tübingen, Germany **2** Tübingen AI Center, Germany ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Mojtaba Barzegari

Reviewers:

- @haniyeka
- @sherbold
- @stsievert

Submitted: 22 September 2023
Published: unpublished

Summary

The cblearn package implements comparison-based machine learning algorithms and routines to process comparison-based data in Python. Comparison-based learning algorithms are used when only comparisons of similarity between data points are available, but no explicit similarity scores or features. For example, humans struggle to assign *numeric* similarities to apples, pears, and bananas. Still, they can easily *compare* the similarity of pears and apples with the similarity of apples and bananas— pears and apples usually appear more similar. There exist comparison-based algorithms for most machine learning tasks, like clustering, regression, or classification (e.g., Balcan et al., 2016; Heikinheimo & Ukkonen, 2013; Perrot et al., 2020); The most frequently applied algorithms, however, are the so-called ordinal embedding algorithms (e.g., Agarwal et al., 2007; Amid & Ukkonen, 2015; Anderton & Aslam, 2019; Ghosh et al., 2019; Maaten & Weinberger, 2012; Tamuz et al., 2011; Terada & Luxburg, 2014). Ordinal embedding algorithms estimate a metric representation, such that the distances between embedded objects reflect the similarity comparisons. These embedding algorithms have recently come into fashion in psychology and cognitive science to objectively quantify the perceived similarity of various stimuli (e.g., Haghiri et al., 2020; Roads & Mozer, 2019; Wills et al., 2009).

Statement of need

This work presents cblearn, an open-source Python package for comparison-based learning. In contrast to related packages, cblearn provides not just a specific algorithm but an ecosystem for comparison-based data with access to multiple real-world datasets and a collection of algorithm implementations. cblearn is fast and user-friendly for applications but flexible for research on new algorithms and methods. The package integrates well into the scientific Python ecosystem; for example, third-party functions for cross-validation or hyperparameter tuning of scikit-learn estimators can typically be used with cblearn estimators. Although our package is relatively new, it has already been used for algorithm development (Mandal et al., 2023) and data analysis in several studies (Assen & Pont, 2022; Künstle et al., 2022; Schönmann et al., 2022; Zhao et al., 2023).

We designed cblearn as a modular package with functions for processing and converting the comparison data in all its varieties (cblearn.preprocessing, cblearn.utils, cblearn.metrics), routines to generate artificial or load real-world datasets (cblearn.datasets), and algorithms for ordinal embedding and clustering (cblearn.embedding, cblearn.cluster).

Various data formats supported

The atomic datum in comparison-based learning is the quadruplet, a comparison of the similarity δ between two pairs (i, j) and (k, l), for example, asserting that $\delta(i, j) < \delta(k, l)$. Alternative comparisons like the triplet, where $i == l$, can be reduced to one or more quadruplets. Comparison-based learning algorithms estimate classes, clusters, or metrics to fulfill as many quadruplets as possible. In ordinal embedding, for example, the problem is to find $x_i, x_j, x_k, x_l \in \mathbb{R}^d$ s.t. $\|x_i - x_j\|_2 < \|x_k - x_l\|_2 \Leftrightarrow \delta(i,j) < \delta(k,l)$.

Besides triplets and quadruplets, there are many ways to ask for comparisons. Some tasks ask for the "odd-one-out", the "most-central" object, or the two most similar objects to a reference. `cblearn` can load these different queries and convert them to triplets, ready for subsequent embedding or clustering tasks.

Different data types can store triplets, and `cblearn` converts them internally. A 2D array with three columns for the object indices (i, j, k) stores a triplet per row. In some applications, it is comfortable to separate the comparison "question" and "response", which leads to an additional list of labels 1, if $\delta(i,j) \leq \delta(i,k)$, and -1, if $\delta(i,j) > \delta(i,k)$. An alternative format stores triplets as a 3-dimensional sparse array. These sparse arrays convert fast back and forth to dense 2D arrays while providing an intuitive comparison representation via multidimensional indexing. For example, the identical triplet can be represented as `[[i, j, k]]`, `([[i, k, j]], [-1])` or `sparse_arr[i, j, k] == 1`.

Interfaces to diverse datasets

There is no Iris, CIFAR, or ImageNet in comparison-based learning—the community lacks accessible real-world datasets to evaluate new algorithms. `cblearn` provides access to various real-world datasets, summarized in Figure 1, with functions to download and load the comparisons. These datasets—typically comparisons between images or words—consist of human responses. Additionally, our package provides preprocessing functions to convert different comparisons to triplets or quadruplets, which many algorithms expect.

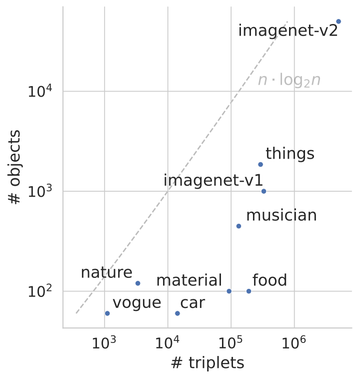

Figure 1: Real-world datasets that can be accessed with `cblearn` cover many object and triplet numbers. Please find a detailed description and references to the dataset authors in our package documentation.

Algorithms implemented for CPU and GPU

In the current version `0.1.2`, `cblearn` implements an extensive palette of ordinal embedding algorithms and a clustering algorithm (Table 1); additional algorithms can be contributed easily within the modular design. Most algorithm implementations use `scipy` to be fast and lightweight. Inspired by the work of Vankadara et al. (2021), we added GPU implementations using *pytorch*, which use the stochastic optimization routines known from deep-learning algorithms. These GPU implementations can be used with large datasets and rapidly adapted thanks to automated differentiation.

Table 1: Algorithm implementations in cblearn. Most of these come in multiple variants: Different backends for small datasets on CPU or large datasets on GPU, or varied objective functions.

Algorithm	Reference
Crowd Kernel Learning	Tamuz et al. (2011)
Fast Ordinal Triplet Embedding	Jain et al. (2016)
Generalized Non-metric MDS	Agarwal et al. (2007)
Maximum-likelihood Difference Scaling	Maloney & Yang (2003)
Soft Ordinal Embedding	Terada & Luxburg (2014)
Ordinal Embedding Neural Network	Vankadara et al. (2021)
Stochastic Triplet Embedding	Maaten & Weinberger (2012)
ComparisonHC (clustering)	Perrot et al. (2020)

User-friendly and compatible API

One of Python's greatest strengths is the scientific ecosystem, into which `cblearn` integrates. Our package does not only make use of this ecosystem internally but adapts their API conventions——every user of `scikit-learn` (Buitinck et al., 2013; Pedregosa et al., 2011) is already familiar with the API of `cblearn`: `numpy` arrays (Harris et al., 2020) are used to store comparison datasets and estimated embeddings; estimator objects use the well-known `scikit-learn` methods `.fit(X, y)`,`.transform(X)`, and `.predict(X)`. This API allows using `scikit-learn` or third-party routines with `cblearn`'s estimators. Interested readers can find a code example in the Supplementary Material, which shows in just four lines how to fetch a real-world dataset, preprocess the data, estimate an embedding, and cross-validate the fit. More examples are available in the package's documentation.

Related work and empirical comparison

Most comparison-based learning algorithms were implemented independently as part of a research paper (e.g., Ghoshdastidar et al., 2019; Hebart et al., 2020; Maaten & Weinberger, 2012; Roads & Mozer, 2019); Just a few of these implementations, for example `loe` (Terada & Luxburg, 2014) or `psiz` (Roads & Mozer, 2019), come in the form of software packages. However, we are unaware of packages that provide an ecosystem for comparison-based learning with various algorithms, datasets, automated-tests, and *scikit-learn*-integration, like `cblearn`.

A small empirical comparison to third-party packages reveals that `cblearn`'s algorithm implementations typically are accurate and fast. Details are described in Supplementary Material. A more comprehensive evaluation of various ordinal embedding algorithms per se, focusing on large data sets, can be found in Vankadara et al. (2021).

Acknowledgements

We want to thank Debarghya Ghoshdastidar, Leena Vankadara, Siavash Haghiri, Michael Lohaus, and especially Michaël Perrot for the inspiring discussions about comparison-based learning in general and the `cblearn` package in particular. Additionally, we thank Thomas Klein for the helpful feedback on this manuscript and Alexander Conzelmann for the contributions to the `cblearn.cluster` module.

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC number 2064/1 – Project number 390727645 and supported by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting David-Elias Künstle.

References

Agarwal, S., Wills, J., Cayton, L., Lanckriet, G., Kriegman, D., & Belongie, S. (2007). Generalized non-metric multidimensional scaling. *Artificial Intelligence and Statistics (AISTATS)*.

Amid, E., & Ukkonen, A. (2015). Multiview triplet embedding: Learning attributes in multiple maps. In F. Bach & D. Blei (Eds.), *Proceedings of the 32nd international conference on machine learning* (Vol. 37, pp. 1472–1480). PMLR.

Anderton, J., & Aslam, J. (2019). Scaling up ordinal embedding: A landmark approach. *International Conference on Machine Learning*, 282–290.

Assen, J. J. R. van, & Pont, S. C. (2022). Identifying the behavioural cues of collective flow perception. *Journal of Vision*, *22*(14), 3985. https://doi.org/10.1167/jov.22.14.3985

Balcan, M.-F., Vitercik, E., & White, C. (2016). Learning combinatorial functions from pairwise comparisons. *Conference on Learning Theory*, 310–335.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., Vanderplas, J., Joly, A., Holt, B., & Varoquaux, G. (2013). API design for machine learning software: Experiences from the scikit-learn project. *arXiv:1309.0238 [Cs.LG]*. https://doi.org/10.48550/arXiv.1309.0238

Ghosh, N., Chen, Y., & Yue, Y. (2019). Landmark ordinal embedding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 32). Curran Associates, Inc.

Ghoshdastidar, D., Perrot, M., & Luxburg, U. von. (2019). Foundations of Comparison-Based Hierarchical Clustering. *Advances in Neural Information Processing Systems (NeurIPS)*.

Haghiri, S., Wichmann, F. A., & Luxburg, U. von. (2020). Estimation of perceptual scales using ordinal embedding. *Journal of Vision*, *20*(9), 14. https://doi.org/10.1167/jov.20.9.14

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hebart, M. N., Zheng, C. Y., Pereira, F., & Baker, C. I. (2020). Revealing the multidimensional mental representations of natural objects underlying human similarity judgements. *Nature Human Behaviour*, *4*(11), 1173–1185. https://doi.org/10.1038/s41562-020-00951-3

Heikinheimo, H., & Ukkonen, A. (2013). The crowd-median algorithm. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, *1*, 69–77. https://doi.org/10.1609/hcomp.v1i1.13079

Jain, L., Jamieson, K. G., & Nowak, R. (2016). Finite Sample Prediction and Recovery Bounds for Ordinal Embedding. *Advances in Neural Information Processing Systems (NeurIPS)*.

Künstle, D.-E., Luxburg, U. von, & Wichmann, F. A. (2022). Estimating the perceived dimension of psychophysical stimuli using triplet accuracy and hypothesis testing. *Journal of Vision*, *22*(13), 5. https://doi.org/10.1167/jov.22.13.5

Maaten, L. van der, & Weinberger, K. (2012). Stochastic triplet embedding. *International Workshop on Machine Learning for Signal Processing*, 1–6. https://doi.org/10.1109/MLSP.2012.6349720

Maloney, L. T., & Yang, J. N. (2003). Maximum likelihood difference scaling. *Journal of Vision*, *3*(8), 5–5. https://doi.org/10.1167/3.8.5

Mandal, A., Perrot, M., & Ghoshdastidar, D. (2023). *A Revenue Function for Comparison-Based Hierarchical Clustering* (No. arXiv:2211.16459). arXiv. https://doi.org/10.48550/arXiv.2211.16459

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research (JMLR)*, *12*(85), 2825–2830.

Perrot, M., Esser, P., & Ghoshdastidar, D. (2020). Near-optimal comparison based clustering. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems (NeurIPS)*.

Roads, B. D., & Mozer, M. C. (2019). Obtaining psychological embeddings through joint kernel and metric learning. *Behavior Research Methods*, *51*(5), 2180–2193. https://doi.org/10.3758/s13428-019-01285-3

Schönmann, I., Künstle, D.-E., & Wichmann, F. A. (2022). Using an Odd-One-Out Design Affects Consistency, Agreement and Decision Criteria in Similarity Judgement Tasks Involving Natural Images. *Journal of Vision*, *22*(14), 3232. https://doi.org/10.1167/jov.22.14.3232

Tamuz, O., Liu, C., Belongie, S., Shamir, O., & Kalai, A. T. (2011). Adaptively learning the crowd kernel. *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML)*.

Terada, Y., & Luxburg, U. (2014). Local ordinal embedding. *International Conference on Machine Learning (ICML)*.

Vankadara, L. C., Haghiri, S., Lohaus, M., Wahab, F. U., & Luxburg, U. von. (2021). Insights into Ordinal Embedding Algorithms: A Systematic Evaluation. *arXiv:1912.01666 [Cs, Stat]*. https://doi.org/10.48550/arXiv.1912.01666

Wills, J., Agarwal, S., Kriegman, D., & Belongie, S. (2009). Toward a perceptual space for gloss. *ACM Transactions on Graphics*, *28*(4), 1–15. https://doi.org/10.1145/1559755.1559760

Zhao, Y., de Ridder, H., Stumpel, J., & Wijntjes, M. (2023). Perceiving style at different levels of information. *Journal of Vision*, *23*(9), 5388. https://doi.org/10.1167/jov.23.9.5388