The Journal of Open Source Software

Choice-Learn: Large-scale choice modeling for operational contexts through the lens of machine learning

Vincent Auriau [1,2¶], **Ali Aouad** [3], **Antoine Désir** [4], **and Emmanuel Malherbe** [2]

1 CentraleSupélec, Université Paris-Saclay, France **2** Artefact Research Center, France **3** London Business School, United Kingdom **4** INSEAD, France ¶ Corresponding author

DOI: 10.21105/joss.06899

Software

- Review
- Repository
- Archive

Editor: Chris Vernon

Reviewers:

- @tmigot
- @samuelduchesne

Submitted: 13 June 2024
Published: 10 September 2024

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

Introduction

Discrete choice models aim at predicting choice decisions made by individuals from a menu of alternatives, called an assortment. Well-known use cases include predicting a commuter's choice of transportation mode or a customer's purchases. Choice models are able to handle assortment variations, when some alternatives become unavailable or when their features change in different contexts. This adaptability to different scenarios allows these models to be used as inputs for optimization problems, including assortment planning or pricing.

Choice-Learn provides a modular suite of choice modeling tools for practitioners and academic researchers to process choice data, and then formulate, estimate and operationalize choice models. The library is structured into two levels of usage, as illustrated in Figure 1. The higher-level is designed for fast and easy implementation and the lower-level enables more advanced parameterizations. This structure, inspired by Keras' different endpoints (Chollet et al., 2015), enables a user-friendly interface. Choice-Learn is designed with the following objectives:

- **Streamlined:** The processing of datasets and the estimation of standard choice models are facilitated by a simple code signature that are consistent with mainstream machine learning packages such as scikit-learn (Pedregosa et al., 2011).
- **Scalable:** Optimized processes are implemented for data storage and models estimation, allowing the use of large datasets and models with a large number of parameters.
- **Flexible:** The codebase can be customized to fit different use cases.
- **Models Library:** The same package provides implementations of both standard choice models and machine learning-based methods, including neural networks.
- **Downstream operations:** Post-processing tools that leverage choice models for assortment planning and pricing are integrated into the library.

The main contributions are summarized in Tables 1 and 2.

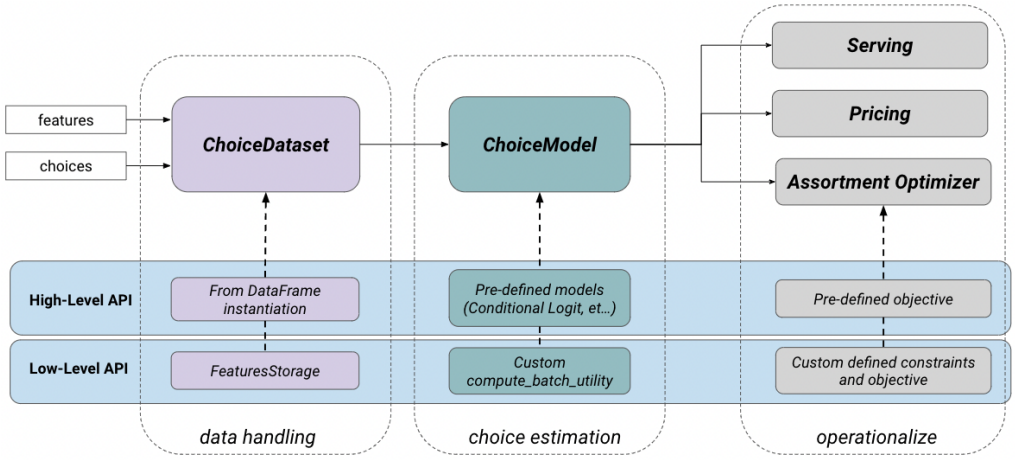


Figure 1: General Organization of the package.

Table 1: Comparison of the different packages for data handling and downstream opereations.

Package	Data Format	Data Batching	Assortment	Pricing
Biogeme	wide	×	×	×
PyLogit	long	×	×	×
Torch-Choice	Multi Index	✓	×	×
Choice-Learn	Features Storage	✓	✓	✓

Table 2: Comparison of the different packages for modelization. CondL, NestL, MixL, and LatC respectively indicate the Conditional Logit, Nested Logit, Mixed Logit and Latent Class models.

Package	Traditional Models	NeuralNet Models	Custom Models	Non-Stochastic Optimizer	Stochastic Optimizer
Biogeme	CondL, NestL,MixL, LatC & more	×	✓	Newton BFGS	×
PyLogit	CondL, NestL,MixL, Asymmetric	×	×	BFGS	×
Torch-Choice	CondL, NestL	×	×	L-BFGS	✓
Choice-Learn	CondL, NestL, LatC	✓	✓	L-BFGS	✓

Statement of need

Data and model scalability

`Choice-Learn`'s data management relies on NumPy (Harris et al., 2020) with the objective of limiting the memory footprint. It minimizes the repetition of items or customers features and defers the jointure of the full data structure until processing batches of data. The package introduces the *FeaturesStorage* object, illustrated in Figure 2, that allows feature values to be referenced only by their ID. These values are substituted to the ID placeholder on the fly in the batching process. For instance, supermarkets features such as surface or position are often stationary. Thus, they can be stored in an auxiliary data structure and in the main dataset, the store where the choice is recorded is only referenced with its ID.

The package stands on Tensorflow (Abadi et al., 2015) for model estimation, offering the possibility to use fast quasi-Newton optimization algorithm such as L-BFGS (Nocedal & Wright, 2006) as well as various gradient-descent optimizers (Kingma & Ba, 2017; Tieleman & Hinton, 2012) specialized in handling batches of data. GPU usage is also possible, which can prove to be time-saving. Finally, the TensorFlow backbone ensures an efficient usage in a production environment, for instance within an assortment recommendation software, through deployment and serving tools, such as TFLite and TFServing.

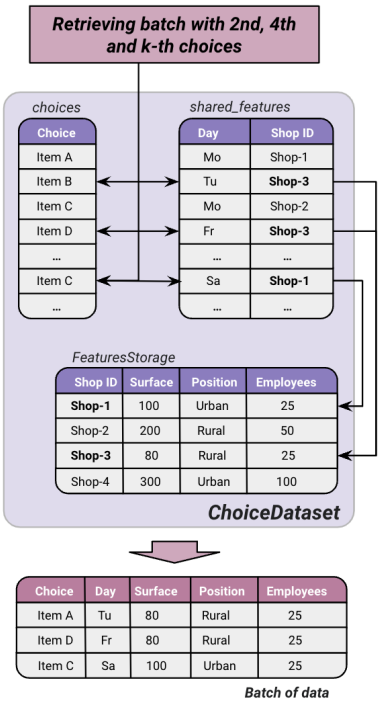


Figure 2: Functioning of the *FeaturesStorage*.

Flexible usage: From linear utility to customized specification

Choice models following the *Random Utility Maximization* principle (McFadden & Train, 2000) define the utility of an alternative $i \in \mathcal{A}$ as the sum of a deterministic part $U(i)$ and a random error ϵ_i. If the terms $(\epsilon_i)_{i \in \mathcal{A}}$ are assumed to be independent and Gumbel-distributed, the probability to choose alternative i can be written as the softmax normalization over the available alternatives $j \in \mathcal{A}$:

$$\mathbb{P}(i|\mathcal{A}) = \frac{e^{U(i)}}{\sum_{j \in \mathcal{A}} e^{U(j)}}$$

The choice-modeler's job is to formulate an adequate utility function $U(.)$ depending on the context. In `Choice-Learn`, the user can parametrize predefined models or freely specify a custom utility function. To declare a custom model, one needs to inherit the *ChoiceModel* class and overwrite the `compute_batch_utility` method as shown in the documentation.

Library of traditional random utility models and machine learning-based models

Traditional parametric choice models, including the Conditional Logit (Train et al., 1987), often specify the utility function in a linear form. This provides interpretable coefficients

but limits the predictive power of the model. Recent works propose the estimation of more complex models, with neural networks approaches (Aouad & Désir, 2022; Han et al., 2022) and tree-based models (Aouad et al., 2023; Salvadé & Hillel, 2024). While existing choice libraries (Bierlaire, 2023; Brathwaite & Walker, 2018; Du et al., 2023) are often not designed to integrate such machine learning-based approaches, `Choice-Learn` proposes a collection including both types of models.

Downstream operations: Assortment and pricing optimization

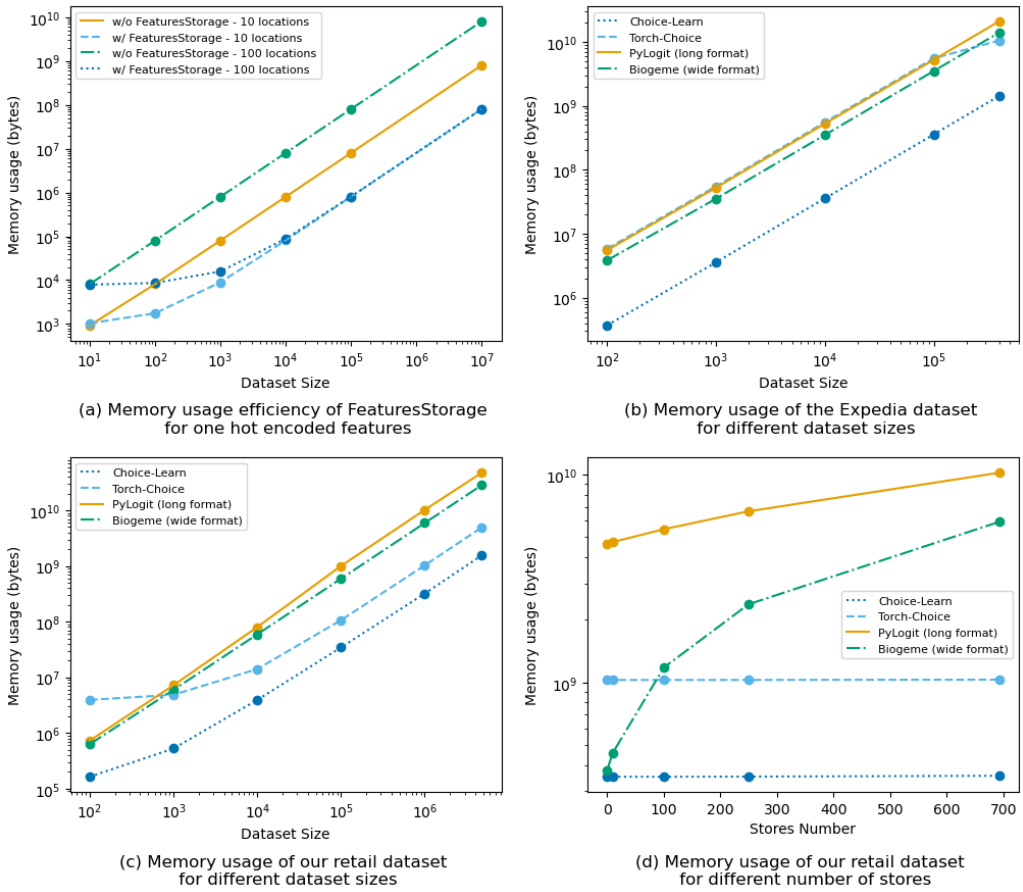
`Choice-Learn` offers additional tools for downstream operations, that are not usually integrated in choice modeling libraries. In particular, assortment optimization is a common use case that leverages a choice model to determine the optimal subset of alternatives to offer customers maximizing a certain objective, such as the expected revenue, conversion rate, or social welfare. This framework captures a variety of applications such as assortment planning, display location optimization, and pricing. We provide implementations based on the mixed-integer programming formulation described in (Méndez-Díaz et al., 2014), with the option to choose the solver between Gurobi (Gurobi Optimization, LLC, 2023) and OR-Tools (Perron & Furnon, 2024).

Memory usage: a case study

We provide in Figure 3 (a) numerical examples of memory usage to showcase the efficiency of the *FeaturesStorage*. We consider a feature repeated in a dataset, such as a one-hot encoding for locations, represented by a matrix of shape (*#locations, #locations*) where each row refers to one location.

We compare four data handling methods on the Expedia dataset (Ben Hamner et al., 2013): pandas.DataFrames (The pandas development team, 2020) in long and wide format, both used in choice modeling packages, Torch-Choice and `Choice-Learn`. Figure 3 (b) shows the results for various sample sizes.

Finally, in Figure 3 (c) and (d), we observe memory usage gains on a proprietary dataset in brick-and-mortar retailing consisting of the aggregation of more than 4 billion purchases in Konzum supermarkets in Croatia. Focusing on the *coffee* subcategory, the dataset specifies, for each purchase, which products were available, their prices, as well as a one-hot representation of the supermarket.



(a) Memory usage efficiency of FeaturesStorage for one hot encoded features

(b) Memory usage of the Expedia dataset for different dataset sizes

(c) Memory usage of our retail dataset for different dataset sizes

(d) Memory usage of our retail dataset for different number of stores

Figure 3: Memory usage experiments.

Acknowledgments

The authors thank Fortenova[1] and Martin Možina for their helpful collaboration and provision of the proprietary dataset.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jozefowicz, R., Jia, Y., Kaiser, L., Kudlur, M., ... Zheng, X. (2015). *TensorFlow, Large-scale machine learning on heterogeneous systems*. https://doi.org/10.5281/zenodo.4724125

Aouad, A., & Désir, A. (2022). Representing random utility choice models with neural networks. *arXiv Preprint arXiv:2207.12877*. https://doi.org/10.48550/arXiv.2207.12877

Aouad, A., Elmachtoub, A. N., Ferreira, K. J., & McNellis, R. (2023). Market segmentation trees. *Manufacturing & Service Operations Management*, *25*(2), 648–667. https://doi.org/10.1287/msom.2023.1195

Ben Hamner, A., Friedman, D., & SSA_Expedia. (2013). Personalize expedia hotel searches - ICDM 2013. *Kaggle*. https://www.kaggle.com/c/expedia-personalized-sort

[1] https://fortenova.hr/en/home/

Bierlaire, M. (2023). *A short introduction to Biogeme* (Technical Report TRANSP-OR 230620). Transport; Mobility Laboratory, Ecole Polytechnique Fédérale de Lausanne.

Brathwaite, T., & Walker, J. L. (2018). Asymmetric, closed-form, finite-parameter models of multinomial choice. *Journal of Choice Modelling*, *29*, 78–112. https://doi.org/10.1016/j.jocm.2018.01.002

Chollet et al. (2015). *Keras*. https://keras.io.

Du, T., Kanodia, A., & Athey, S. (2023). Torch-choice: A PyTorch package for large-scale choice modelling with python. *arXiv Preprint arXiv:2304.01906*. https://doi.org/10.48550/arXiv.2304.01906

Gurobi Optimization, LLC. (2023). *Gurobi Optimizer Reference Manual*. https://www.gurobi.com.

Han, Y., Pereira, F. C., Ben-Akiva, M., & Zegras, C. (2022). A neural-embedded discrete choice model: Learning taste representation with strengthened interpretability. *Transportation Research Part B: Methodological*, *163*, 166–186. https://doi.org/10.1016/j.trb.2022.07.001

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.

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization. *arXiv Preprint arXiv:1412.6980*. https://doi.org/10.48550/arXiv.1412.6980

McFadden, D., & Train, K. (2000). Mixed MNL models for discrete response. *Journal of Applied Econometrics*, *15*(5), 447–470. https://doi.org/10.1002/1099-1255(200009/10)15:5%3C447::aid-jae570%3E3.3.co;2-t

Méndez-Díaz, I., Miranda-Bront, J. J., Vulcano, G., & Zabala, P. (2014). A branch-and-cut algorithm for the latent-class logit assortment problem. *Discrete Applied Mathematics*, *164*, 246–263. https://doi.org/10.1016/j.dam.2012.03.003

Nocedal, J., & Wright, S. J. (2006). Large-scale unconstrained optimization. In *Numerical optimization* (pp. 164–192). Springer New York. https://doi.org/10.1007/0-306-48332-7_250

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, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Perron, L., & Furnon, V. (2024). *OR-tools* (Version v9.9). Google; https://developers.google.com/optimization/.

Salvadé, N., & Hillel, T. (2024). RUMBoost: Gradient boosted random utility models. *arXiv Preprint arXiv:2401.11954*. https://doi.org/10.48550/arXiv.2401.11954

The pandas development team. (2020). *Pandas-dev/pandas: Pandas*. Zenodo. https://doi.org/10.5281/zenodo.3509134

Tieleman, T., & Hinton, G. (2012). Lecture 6.5 RMSProp, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*, *6*.

Train, K. E., McFadden, D. L., & Ben-Akiva, M. (1987). The demand for local telephone service: A fully discrete model of residential calling patterns and service choices. *The RAND Journal of Economics*, *18*(1), 109–123. https://doi.org/10.2307/2555538