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

Vincent Auriau[A,B], Ali Aouad[C], Antoine Désir[D] and Emmanuel Malherbe[B]
[A]CentraleSupélec, Université Paris-Saclay, France
[B]Artefact Research Center, France
[C]London Business School, United Kingdom
[D]INSEAD, France

1. Introduction

Discrete choice models aim at predicting choice decisions made by individuals from a menu of alternatives, called an assortment[cite: 1]. Well-known use cases include predicting a commuter's choice of transportation mode or a customer's purchases[cite: 1]. Choice models are able to handle assortment variations, when some alternatives become unavailable or when their features change in different contexts[cite: 1]. This adaptability to different scenarios allows these models to be used as inputs for optimization problems, including assortment planning or pricing[cite: 1].

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[cite: 1]. The library is structured into two levels of usage, as illustrated in Figure 1[cite: 1]. The higher-level is designed for fast and easy implementation and the lower-level enables more advanced parameterizations[cite: 1]. This structure, inspired by Keras' different endpoints [7], enables a user-friendly interface[cite: 1]. Choice-Learn is designed with the following objectives[cite: 1]:

- **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 [16][cite: 1].
- **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[cite: 1].
- **Flexible:** The codebase can be customized to fit different use cases[cite: 1].
- **Models Library:** The same package provides implementations of both standard choice models and machine learning-based methods, including neural networks[cite: 1].
- **Downstream operations:** Post-processing tools that leverage choice models for assortment planning and pricing are integrated into the library[cite: 1].

The main contributions are summarized in Tables 1 and 2.

2. Statement of need

2.1. Data and model scalability

`Choice-Learn`'s data management relies on NumPy [11] with the objective of limiting the memory footprint[cite: 1]. It minimizes the repetition of items or customers features and defers the jointure

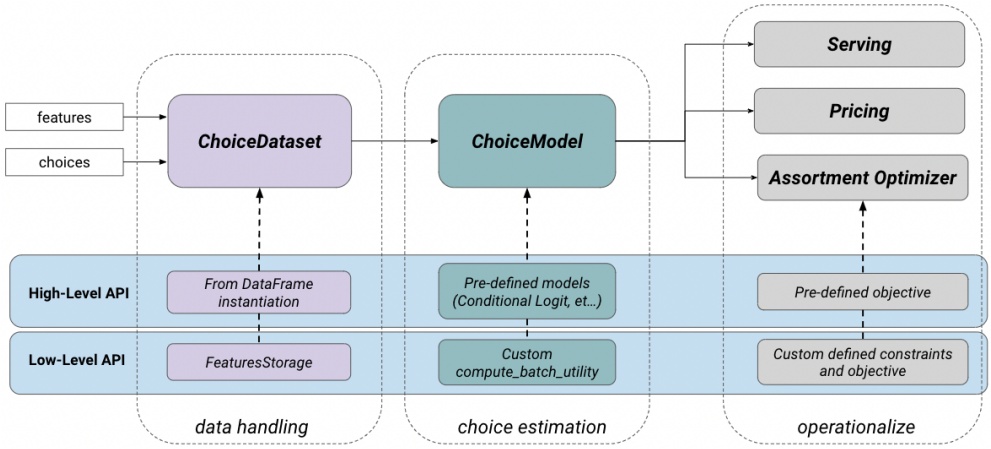


Figure 1 | General Organization of the package.

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	✓	✓	✓

of the full data structure until processing batches of data[cite: 1]. The package introduces the *FeaturesStorage* object, illustrated in Figure 2, that allows feature values to be referenced only by their ID[cite: 1]. These values are substituted to the ID placeholder on the fly in the batching process[cite: 1]. For instance, supermarkets features such as surface or position are often stationary[cite: 1]. 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[cite: 1].

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	✓

The package stands on Tensorflow [1] for model estimation, offering the possibility to use fast quasi-Newton optimization algorithm such as L-BFGS [15] as well as various gradient-descent optimizers [20, 12] specialized in handling batches of data[cite: 1]. GPU usage is also possible, which can prove to be time-saving[cite: 1]. 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[cite: 1].

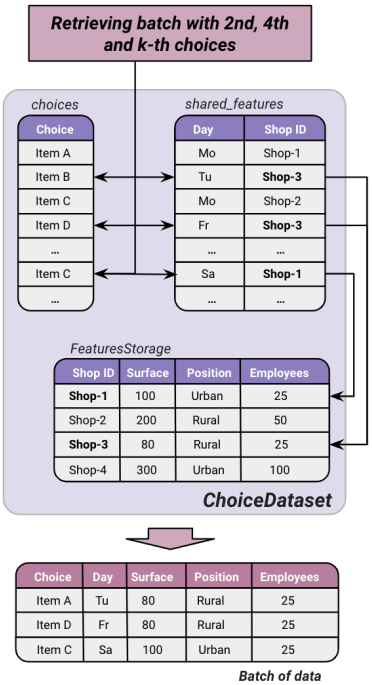


Figure 2 | Functioning of the *FeaturesStorage*.

2.2. Flexible usage: From linear utility to customized specification

Choice models following the *Random Utility Maximization* principle [13] 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[cite: 1]. 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 $j \in \mathcal{A}$[cite: 1]:

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

The choice-modeler's job is to formulate an adequate utility function $U(.)$ depending on the context[cite: 1]. In `Choice-Learn`, the user can parametrize predefined models or freely specify a custom utility function[cite: 1]. To declare a custom model, one needs to inherit the *ChoiceModel* class and overwrite the `compute_batch_utility` method as shown in the documentation[cite: 1].

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

Traditional parametric choice models, including the Conditional Logit [21], often specify the utility function in a linear form[cite: 1]. This provides interpretable coefficients but limits the predictive power of the model[cite: 1]. Recent works propose the estimation of more complex models, with neural networks approaches [10, 2] and tree-based models [18, 3][cite: 1]. While existing choice libraries [5, 6, 8] are often not designed to integrate such machine learning-based approaches, `Choice-Learn` proposes a collection including both types of models[cite: 1].

2.4. Downstream operations: Assortment and pricing optimization

`Choice-Learn` offers additional tools for downstream operations, that are not usually integrated in choice modeling libraries[cite: 1]. 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[cite: 1]. This framework captures a variety of applications such as assortment planning, display location optimization, and pricing[cite: 1]. We provide implementations based on the mixed-integer programming formulation described in [14], with the option to choose the solver between Gurobi [9] and OR-Tools [17][cite: 1].

3. Memory usage: a case study

We provide in Figure 3 (a) numerical examples of memory usage to showcase the efficiency of the *FeaturesStorage*[cite: 1]. 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[cite: 1].

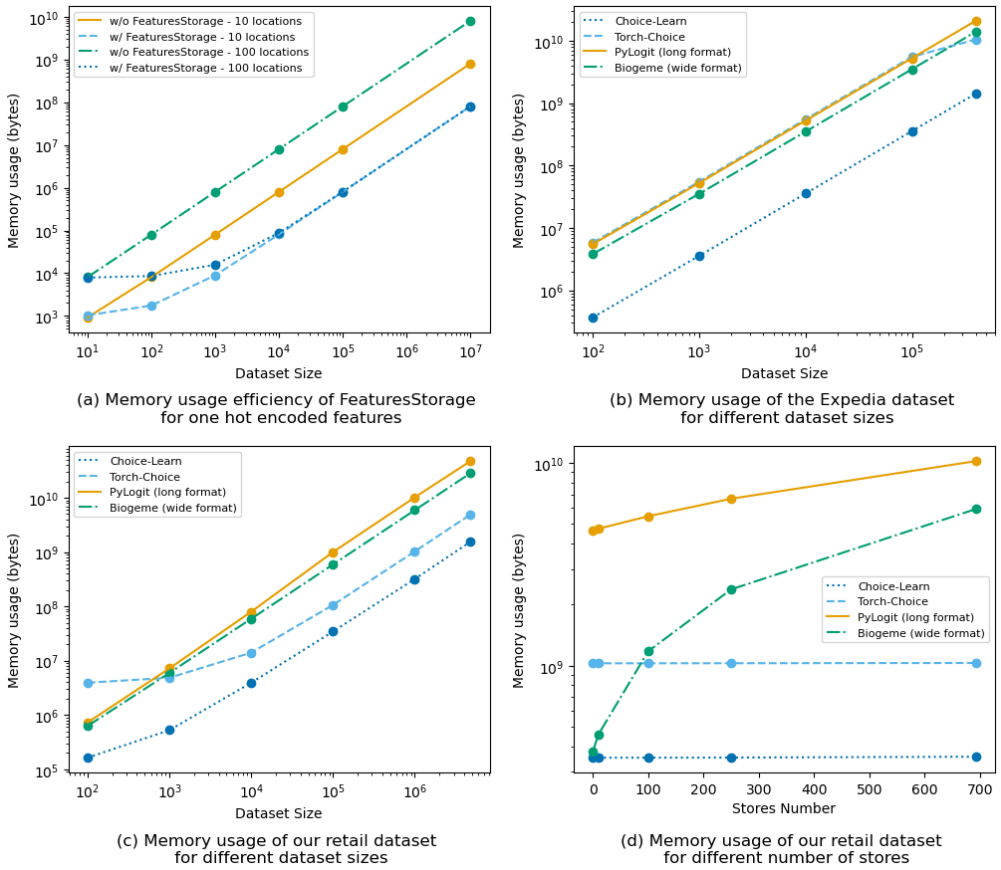


Figure 3 | Memory usage experiments.

We compare four data handling methods on the Expedia dataset [4]: pandas.DataFrames [19] in long and wide format, both used in choice modeling packages, Torch-Choice and `Choice-Learn`[cite: 1]. Figure 3 (b) shows the results for various sample sizes[cite: 1].

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[cite: 1]. 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[cite: 1].

4. Acknowledgments

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

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow, Large-scale machine learning on heterogeneous systems, November 2015.

[2] Ali Aouad and Antoine Désir. Representing random utility choice models with neural networks. *arXiv preprint arXiv:2207.12877*, 2022.

[3] Ali Aouad, Adam N Elmachtoub, Kris J Ferreira, and Ryan McNellis. Market segmentation trees. *Manufacturing & Service Operations Management*, 25(2):648–667, 2023.

[4] Adam Ben Hamner, Dan Friedman, and $SSA_{E}xpedia$. $Personalizeexpediahotelsearches$ – $icdm2013.Kaggle$, 2013.

[5] Michel Bierlaire. A short introduction to Biogeme. Technical Report TRANSP-OR 230620, Transport and Mobility Laboratory, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2023.

[6] Timothy Brathwaite and Joan L. Walker. Asymmetric, closed-form, finite-parameter models of multinomial choice. *Journal of Choice Modelling*, 29:78–112, December 2018.

[7] Chollet et al. Keras. https://keras.io, 2015.

[8] Tianyu Du, Ayush Kanodia, and Susan Athey. Torch-choice: A pytorch package for large-scale choice modelling with python. *arXiv preprint arXiv:2304.01906*, 2023.

[9] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual. https://www.gurobi.com, 2023.

[10] Yafei Han, Francisco Camara Pereira, Moshe Ben-Akiva, and Christopher Zegras. A neural-embedded discrete choice model: Learning taste representation with strengthened interpretability. *Transportation Research Part B: Methodological*, 163:166–186, 2022.

[11] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del

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

Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020.

[12] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2017.

[13] Daniel McFadden and Kenneth Train. Mixed mnl models for discrete response. *Journal of applied Econometrics*, 15(5):447–470, 2000.

[14] Isabel Méndez-Díaz, Juan José Miranda-Bront, Gustavo Vulcano, and Paula Zabala. A branch-and-cut algorithm for the latent-class logit assortment problem. *Discrete Applied Mathematics*, 164:246–263, 2014. Combinatorial Optimization.

[15] Jorge Nocedal and Stephen J. Wright. *Large-Scale Unconstrained Optimization*, pages 164–192. Springer New York, New York, NY, 2006.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[17] Laurent Perron and Vincent Furnon. Or-tools. https://developers.google.com/optimization/.

[18] Nicolas Salvadé and Tim Hillel. Rumboost: Gradient boosted random utility models. *arXiv preprint arXiv:2401.11954*, 2024.

[19] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[20] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5 RMSProp, coursera: Neural networks for machine learning. *University of Toronto, Technical Report*, 6, 2012.

[21] Kenneth E. Train, Daniel L. McFadden, and Moshe Ben-Akiva. 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, 1987.