Better Capturing Interactions between Products in Retail: Revisited Negative Sampling for Basket Choice Modeling

Jules Désir[A], Vincent Auriau[A,B], Martin Možina[C] and Emmanuel Malherbe[A]
[A]Artefact Research Center, Paris, France
[B]MICS, CentraleSupélec Université Paris-Saclay, Gif-sur-Yvette, France
[C]Fortenova Group, Zagreb, Croatia

Brick-and-mortar retailers face many different challenges that involve understanding thoroughly its products catalog and customer preferences. In particular, assortment optimization - proposing the ideal mix of products - and promotion planning hold a pivotal role in their strategy. By leveraging sales data, retailers can make informed decisions on which products to sell and how to manage inventory, based on customer preferences as well as regional and seasonal trends. It is especially crucial to capture interactions between products, in order to minimize the number of items that cannibalize each other's sales and to ensure that complementary products, which are often purchased together, are conjointly available and sold across all stores. In this paper, we propose a model of shopping basket that learns embeddings to represent interactions between products, prices and stores. Our model is built to uncover sales patterns from very large transaction datasets. In particular, the optimization loss is computed with random negative samples in order to overcome the computational bottlenecks that arise with large number of items. Our experiments on synthetic data show the efficiency of drawing such negative samples based on the actual assortment of available products, with better results than approaches from the literature. We also validate our approach by training and evaluating our model on a dataset composed of millions of transactions from *Fortenova Group*, one of the leading retail company in Southeast Europe. Our model showcases promising applications in the sector of retail, with enriched interfaces to efficiently support category managers.

1. Introduction

Choosing the optimal combination of products to sell at the right price is a crucial stake for retailers. It is needed to select which products, among the hundreds of thousands offered by various suppliers, should be offered for sale in each store, or should be under promotion. This decision can be taken using insights such as past revenues or current trends.

It is also highly crucial to take into consideration the different products synergies. Indeed, a customer planning to bake a cake should be able to purchase all the necessary ingredients or he may choose to shop at another supermarket. This notion of complementarity between products should be carefully considered, along with cannibalization, where the presence of two similar items lead to a reduction in each other's sales. Historically, this assortment selection has been done manually by the category managers, based on their business experience. With thousands of daily customers in its different stores, a supermarket company typically records hundreds of millions of purchases per year. This data - baskets of co-purchased items - conveys rich patterns, especially compared to assortments which remain relatively stable due to logistical reasons. Gathering statistical insights from these data can greatly help category managers make informed decisions on the assortment selection.

Choice modeling is a common approach that leverages purchase datasets to understand customers preferences. Each product is assigned a utility, a latent value that reflects its probability to be purchased.

Published at Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD) 2025
Artefact Research Center, Paris, France

Common models use a linear representation of utility [2, 10], while more recent work has shown that incorporating non-linear effects can help in better capturing the heterogeneity in the different customer behaviors. In particular, neural networks [6, 8] have shown promising performances. Besides, estimating and leveraging product interactions is a problem yet to be tackled. Indeed, the exponentially growing number of interactions to take into account leads to difficulties in using classical methods [11, 7] with a large number of products. Another approach, the nested logit model [4], requires the modeler to decide a priori the sets of substitutable items, which is on the contrary what we want the model to learn.

In this paper, we propose a neural-network choice model that learns latent representations of items, shops and baskets, with a structure capable of capturing product interactions from a large dataset of purchase receipts. It enables to interpret the choice probabilities predicted by the model, and has promising applications for enriching interfaces used by category managers. Such interfaces are used to plan promotion campaigns or select the assortment of a shop and would be more informative if the interactions between products were displayed.

Our contributions in this paper are the following. First, we propose a choice model capturing interactions between products, able to be learned on large-scale data with no or few features to describe items, shops or customers. Second, we propose to leverage the assortment of available products in a soft manner within the negative sampling during the training procedure. Last, we present experiments on synthetic data and a real-world retailer dataset, confirming the benefits of our approach when compared to existing models. Our code is publicly available on GitHub[1].

2. Related Work

Our paper relates to two strands of the literature: discrete choice models and representation learning, with similarities with word embeddings.

2.1. Discrete Choice Models

Discrete choice models try to explain or predict choice decisions made by humans from a set of alternatives, called an assortment. They have been widely adopted for a large range of applications, usually involving a customer decision, such as choice of transportation mode or products purchases. The discrete choice framework based on Random Utility Maximization [10] considers that an individual attributes a utility to the different options and chooses the one that maximizes this utility. The objective of discrete choice modeling is to estimate this latent utility function harnessing data with past choices or preferences. The classical conditional logit model [2] formulates a linear utility function leveraging the different available features of the problem. Recent works use more complex functions that introduce nonlinearities, using for example neural networks [6, 5, 35, 3] or tree-based models [34, 31].

Discrete choice models are designed to handle the variations of options availability within the assortment. This is particularly important in a retail context where products might be out of stock or simply not sold in specific locations. However, they are usually built to model a single choice and still face challenges when modeling bundles of options, like we observe in retail. These challenges result in limited methods for estimating interactions between these choices. Interaction terms can be introduced within the utility function [11, 7], but such formulations are typically restricted to a small number of distinct options. Indeed, the number of possible bundles grows exponentially with

[1]The different Python implementations can be found in the Choice-Learn package at https://github.com/artefactory/choice-learn.

the number of available items, which can be very high in applications such as retail.

2.2. Representation learning

Object representation in machine learning has significantly evolved with the advent of neural networks [19]. Unlike traditional machine learning methods, which could mainly transform pre-existing features, neural networks can learn continuous representations when the data is categorical and comes with no features [20]. Traditionally, such categorical data is represented using one-hot vectors [21] that have the size of the set of possible values. In comparison, learned embeddings provide a continuous representation that replaces these vectors in a generally much lower dimension. One may note that the term "embedding" sometimes refers to a transformed object's features after multiple operations [15], but in this paper it will only refer to categorical data representation.

One of the primary data structures that have greatly benefited from embeddings is word representation [18]. Traditionally, words were represented using sparse vectors [16], similar to weighted one-hot vectors. [9] proved the efficiency to train a neural network where each word in the vocabulary is associated with a learned dense continuous vector. This training process typically requires large-scale data and is often conducted in a self-supervised manner using natural data [14], i.e. an unlabeled corpus of documents. In this case, negative sampling [17] is commonly employed to complement the positive cases derived from the natural data. This involves randomly sampling words from the vocabulary, under the assumption that they are likely to be implausible outputs.

Embeddings have also found applications beyond words. They can be used for any categorical data, typically the labels of a multiclass classification [23, 22], the nodes of a graph [24], or the rows of tabular data represented as graphs [25]. In a scenario similar to choice modeling, collaborative filtering for recommender systems, matrix factorization or neural networks aim at building embeddings for each user and product [26, 27]. In this case, the model is generally trained on the scores given by users on products, such as movies [28]. To our knowledge, limited research has proposed embedding structures for neural network-based choice models. Among existing work, [8] and [36] embeddings are fed directly into the softmax function and thus do not model any interaction. [37] incorporates interaction terms with respect to the items in the assortments, but focuses solely on single-choice modeling, which does not fully leverage the rich information provided by bundles of items purchased together. [1] presents a closer approach to our work; however, their method requires heavy computing since their neural network is Bayesian and they need to generate all subsets of items within a bundle, both during training and inference.

3. Model

3.1. Context and Objectives

We build a model with the following characteristics in mind:

- Only a few or no features are available to characterize the items other than their price and whether they are on promotion. The main signals come from co-purchase patterns.
- The model should capture complementarity and cannibalization between products — i.e. respectively the tendency for items to be purchased together within a single basket and the reduction of sales when items are both available in the assortment.
- The model is designed to efficiently handle large datasets, both in terms of the number of items considered as well as the volume of purchases.

This setup corresponds to our retail scenario and would apply to other sectors with brick-and-mortar

stores.

3.2. Notations

The data we consider consists of a collection of purchases, each representing a basket of items bought by a customer. Additional contextual information includes the store where the purchases took place, the list of items that were available at that time, and their prices. More formally, we write $\mathcal{I}$ the set of items sold in the set $\mathcal{S}$ of stores. The observed purchases correspond to a tuple $(\mathcal{B}, \mathcal{A}, s, p)$ with:

- $\mathcal{B} \subseteq \mathcal{I}$ the basket of purchased items;
- $\mathcal{A} \subseteq \mathcal{I}$ the products assortment available at the time of the purchase (by definition, $\mathcal{B} \subseteq \mathcal{A}$);
- $s \in \mathcal{S}$ the store in which the purchase happened;
- p the prices for all the items in $\mathcal{A}$ at the time of the purchase. In particular, p_i is the price of item i.

We decide not to consider duplicated items in the basket. This choice reflects our objective of capturing interactions between items of the catalog.

3.3. Conditional definition of the utility

We propose a conditional formulation of the utility. Considering an already chosen basket of items $\mathcal{B}$, we write the utility of an additional item as $V(i|\mathcal{B}, s, p)$, modeled as two independent contributions:

$$V(i|\mathcal{B}, s, p) = V^{pref}(i|s, p) + V^{inter}(i|\mathcal{B}) \tag{1}$$

where V^{pref} represents the intrinsic utility (or "preference") attributed to the item and the interaction utility V^{inter} depends on the other items that will also be purchased. V^{pref} can be understood as the popularity of an item, regardless of the other items for sale. In our case, it would only depend on the item, its price and the store where it is purchased. Following classical formulations of the utility, we define V^{pref} as an itemwise intercept, a logarithmic contribution of the price and an elasticity that depends on the item and the store:

$$V^{pref}(i|s, p) = \alpha_i + \boldsymbol{\beta}_i \cdot \boldsymbol{\delta}_s \log p_i \tag{2}$$

where $\cdot$ denotes the scalar product and variables in bold are vectors. We have $\alpha_i \in \mathbb{R}$ and $\boldsymbol{\beta}_i, \boldsymbol{\delta}_s \in \mathbb{R}^K$, with $K \in \mathbb{N}^*$ the size of the latent representation to be chosen. $\boldsymbol{\delta}_s$ captures that each store has a different population with its own elasticities. The formulation of $V^{pref}(i|s, p)$ can be easily modified to introduce multiple underlying factors governing customer choice.

For the interaction term, we use a formulation inspired by methods such as Word2Vec [9], where, in a similar fashion, the probability that a word matches its surrounding words is estimated. In our case, the context is given by items in the basket $\mathcal{B}$, so that V^{inter} is expressed as:

$$V^{inter}(i|\mathcal{B}) = \boldsymbol{\gamma_i} \cdot \boldsymbol{E}_{\mathcal{B}} \tag{3}$$

with $\gamma_i \in \mathbb{R}^K$ a learned representation of item i and $\boldsymbol{E}_{\mathcal{B}}$ the representation for basket $\mathcal{B}$. Different formulations of such basket embedding could be used, we have opted for: $\boldsymbol{E}_{\mathcal{B}} = \frac{1}{|\mathcal{B}|} \sum_{j \in \mathcal{B}} \boldsymbol{\gamma_j}$. One notes that it would be possible to learn different items embeddings for $\boldsymbol{E}_{\mathcal{B}}$ than the $\boldsymbol{\gamma_j}$ ("encoding-decoding" structure).

We assume that the customer behaves rationally and will maximize their utility during a shopping session. Following Discrete Choice Models formulation, we consider that the final utility U is the sum

of the deterministic part V as well as a random contribution, ϵ_i:

$$U(i|\mathcal{B}, s, p) = V(i|\mathcal{B}, s, p) + \epsilon_i$$

from which we can derive probabilities, as described in the following section.

3.4. Purchase probability

The random variable ϵ_i is assumed to be independent across items and to follow a zero-mean Gumbel distribution (generalized type I extreme value). Consequently, the conditional choice probability of choosing an item $i \in \mathcal{A}$ that is not yet in the basket is a softmax over the remaining items in the assortment $\mathcal{A}$:

$$\mathbb{P}(i|\mathcal{B}, s, p) = \mathbb{P}\Big(U\left(i|\mathcal{B}, s, p\right) > U\left(j|\mathcal{B}, s, p\right) \ \ \forall j \in \mathcal{A} \setminus \mathcal{B}, j \neq i\Big) \tag{4}$$

$$= \frac{e^{V(i|\mathcal{B},s,p)}}{\sum_{j\in(\mathcal{A}\setminus\mathcal{B})} e^{V(j|\mathcal{B},s,p)}} \tag{5}$$

where $\mathbb{P}(i|\mathcal{B}, s, p)$ is the probability to add the item i to the basket $\mathcal{B}$ at the store s with the prices p and is estimated with the latent parameters $\Theta = \{\alpha, \boldsymbol{\beta}, \boldsymbol{\delta}, \boldsymbol{\gamma}\}$. We set this probability to zero when $i \notin \mathcal{A}$ or $i \in \mathcal{B}$.

4. Implementation for large-scale data

Given a large dataset of N shopping sessions $\{(\mathcal{B}_k, \mathcal{A}_k, s_k, p_k)\}_{k=1}^N$, our goal is to infer the latent parameters $\Theta \subset \{\alpha, \boldsymbol{\beta}, \boldsymbol{\delta}, \boldsymbol{\gamma}\}$ which are the trainable weights of our model. We now describe how to learn the optimal Θ^* from the dataset.

4.1. Training Procedure

The model parameters Θ^* are estimated by leveraging all baskets data. To this end, similarly to [11] and [29] (Chapter 12), we aim at maximizing the quantity:

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} \prod_{k=1}^{N} \prod_{i\in\mathbb{B}_k} \mathbb{P}(i|\mathcal{B}_k\backslash i, s_k, p_k)$$

which can be seen as a form of self-supervision similar to text, where the missing word is predicted based on the remaining words of its context sentence. One notes that it differs from [1] where they maximize the probability of the baskets, which is not explicitly defined in our model. Besides, for this purpose they generate all possible sub-baskets $\mathcal{B}' \subseteq \mathcal{B}$, which we believe is not always informative and makes computation exponentially costly as the basket size grows.

One also notes that the formula of $\mathbb{P}(i|\mathcal{B}_k\backslash i, s_k, p_k)$ (Equation 5) is computationally heavy too, since $\mathcal{A}$ can be very large. Instead, we normalize $e^{V(i|\mathcal{B},s,p)}$ over a set of randomly drawn negative samples, formally written $\mathcal{NS}$ (whose size $|\mathcal{NS}|$ is a hyperparameter). We thus approximate $\mathbb{P}(i|\mathcal{B}_k\backslash i, s_k, p_k)$ with:

$$f(i, \mathcal{B}_k\backslash i, s_k, p_k) = \frac{e^{V(i|\mathcal{B}_k,s_k,p_k)}}{e^{V(i|\mathcal{B}_k,s_k,p_k)} + \sum_{j\in\mathcal{NS}} e^{V(j|\mathcal{B}_k,s_k,p_k)}}$$

Such negative sampling has also been used in [30] and [9], and can be shown to approximate well the log probabilities of the softmax. This strategy is particularly important in a large-scale environment with thousands of different products.

In practice, we maximize a lower bound of $\prod_{k=1}^{N} \prod_{i \in \mathcal{B}_k} f(i|\mathcal{B}_k \backslash i, s_k, p_k)$:

$$\prod_{k=1}^{N} \prod_{i \in \mathcal{B}_k} f(i|\mathcal{B}_k \backslash i, s_k, p_k) \geq \prod_{k=1}^{N} \prod_{i \in \mathcal{B}_k} \frac{e^{V(i|\mathcal{B}_k, s_k, p_k)}}{(1 + e^{V(i|\mathcal{B}_k, s_k, p_k)}) \prod_{j \in \mathcal{NS}} (1 + e^{V(j|\mathcal{B}_k, s_k, p_k)})}$$

which gives us the final loss to be minimized during our training, that is the opposite of the log of the lower bound:

$$\mathcal{L} = -\sum_{k=1}^{N} \sum_{i \in \mathcal{B}_k} \left(\log \sigma[V(i|\mathcal{B}_k \backslash i, s_k, p_k)] + \sum_{j \in \mathcal{NS}} \log \left(1 - \sigma \left[V \left(j|\mathcal{B}_k \backslash i, s_k, p_k\right)\right]\right) \right) \tag{6}$$

where $\sigma[x] = \frac{e^x}{1+e^x}$ is the sigmoid function. One notes that this loss corresponds to a binary cross-entropy where the $V(i|\mathcal{B}_k \backslash i, s_k, p_k)$ are associated to the positive labels and the negative samples $V(j|\mathcal{B}_k \backslash i, s_k, p_k)$ to the negative labels.

This loss is minimized during a stochastic gradient descent. At each batch of baskets, the negative samples $\mathcal{NS}$ are randomly drawn, with a different sampling for each epoch in the case of multiple epochs on the dataset. In the following, we describe our method to randomly sample $\mathcal{NS}$.

4.2. Negative Sampling

In scenarios involving a fixed set of options, such as the vocabulary in the context of natural language [9], negative samples $\mathcal{NS}$ are uniformly sampled from all other options, excluding the true option. In our choice modeling scenario, however, there are several key differences. First, we exclude items already present in basket $\mathcal{B}_k$, as our context does not allow for duplicated items (see Sections 3.1 and 3.2). Second, we propose to draw negative samples $\mathcal{NS}$ exclusively from items within the available assortment $\mathcal{A}_k$. Formally, we enforce the condition:

$$\mathcal{NS} \subset \mathcal{A}_k \backslash \mathcal{B}_k \tag{7}$$

Implicitly, the negative samples are generated depending on both $\mathcal{A}_k$ and $\mathcal{B}_k$ on the data point k. It could be explicitly written $\mathcal{NS}(\mathcal{A}_k, \mathcal{B}_k)$. Practically, it means that $\mathcal{A}_k$ must be provided as information during the training process. In contrast, [1] draws their negative samples from the entire item catalog outside the considered basket, that is to say $\mathcal{I} \backslash \mathcal{B}_k$.

The rationale for the use of the assortment during negative sampling is that it prevents the model from considering that two items are cannibalizing each other just because they are not available both at the same time in the training data. Indeed, when item $j \in \mathcal{I} \backslash \mathcal{B}_k$ is drawn as a negative sample of item $i \in \mathcal{B}_k$, then the conditional utility $V\left(j|\mathcal{B}_k \backslash i, s_k, p_k\right)$ will be reduced during the gradient descent that minimizes the loss $\mathcal{L}$ (see Equation 6). Imposing item j to be in the assortment $\mathcal{A}_k$ means that such reduction of a conditional probability during the gradient descent will happen only to items that could have been bought with item i but had not, leaving items outside of the assortment untouched and thus enabling the model to learn a difference between items that are in the assortment and items that are not. In other words, drawing a negative sample $j \in \mathcal{I} \backslash \mathcal{B}_k$ that is not in assortment $\mathcal{A}_k$ amounts to assuming that j would have been cannibalizing item i if j were in assortment $\mathcal{A}_k$, which is not necessarily true.

We consider this method of incorporating assortment $\mathcal{A}_k$ in the estimation of $\mathbb{P}(i|\mathcal{B}_k \backslash i, s_k, p_k)$ to be simple yet sufficient and effective, with significant benefits, as demonstrated in the following section.

5. Experiments

We have conducted several types of experiments that validate the relevance of our approach. First, synthetic experiments let us showcase how it behaves on a smaller and controlled environment. Second, we share results obtained on a large real-world industrial dataset issued from Fortenova Group's supermarkets.

5.1. Synthetic Experiments

5.1.1. Dataset

We build a synthetic dataset for which the catalog has 8 items, $\mathcal{I} = \{1, \ldots, 8\}$, with the following interactions:

- **Cannibalization:** $\{1, 2, 3\}$ on the one hand and $\{4, 5, 6\}$ on the other hand form groups of items cannibalizing each other.
- **Complementarity:** each of the items in $\{1, 2, 3\}$ are complementary to each of the items in $\{4, 5, 6\}$;
- **Neutral:** 7 and 8 are neutral in the sense that they don't have specific interaction with other items.

As a concrete example, we can consider that items $\{1, 2, 3\}$ are badminton shuttlecocks, items $\{4, 5, 6\}$ are badminton rackets while items $\{7, 8\}$ are a t-shirt and a short. We implemented a routine that, given an assortment $\mathcal{A} \subseteq \mathcal{I}$, generates a dataset $\mathcal{D}$ of purchased baskets. These baskets are generated following the interactions described above, without price or shop effect (apart from its assortment). The details of this implementation are in Appendix A[2]. In the left of Figure 1, we illustrate the simulated interactions by computing $\mathbb{P}(i \in \mathcal{B} | j \in \mathcal{B})$ on baskets generated by our routine for a full assortment $\mathcal{A} = \mathcal{I}$. A train dataset $\mathcal{D}_{train}$ and a test dataset $\mathcal{D}_{test}$, illustrated in Figure 2, are constituted by sampling baskets for the different following assortments:

- Training set $\mathcal{D}_{train}$:
  - Store 1: $\mathcal{A}_1 = \{1, 2, 4, 5, 7, 8\} = \mathcal{I} \backslash \{3, 6\}$ yields $\mathcal{D}_1$;
  - Store 2: $\mathcal{A}_2 = \{1, 3, 5, 6, 7, 8\} = \mathcal{I} \backslash \{2, 4\}$ yields $\mathcal{D}_2$;
  - Store 3: $\mathcal{A}_3 = \{2, 3, 4, 6, 7, 8\} = \mathcal{I} \backslash \{1, 5\}$ yields $\mathcal{D}_3$;
- Test set $\mathcal{D}_{test}$:
  - Store 4: $\mathcal{A}_4 = \{1, 2, 5, 6, 7, 8\} = \mathcal{I} \backslash \{3, 4\}$ yields $\mathcal{D}_4$;
  - Full Assortment: $\mathcal{A}_{full} = \mathcal{I}$, i.e. all items yields $\mathcal{D}_{full}$.

5.1.2. Protocol

We compare several versions of our approach with other state of the art models:

- Shopper [1]: A choice model that iteratively builds the most probable basket. It models the probability to add an item to an already partially built basket with price and interactions effects. The main difference lies in the expression of the basket probability that is computed by summing over all the possible ordered permutations of the basket.

[2]Appendix can be found in the GitHub repository at https://github.com/artefactory/alea-carta-est

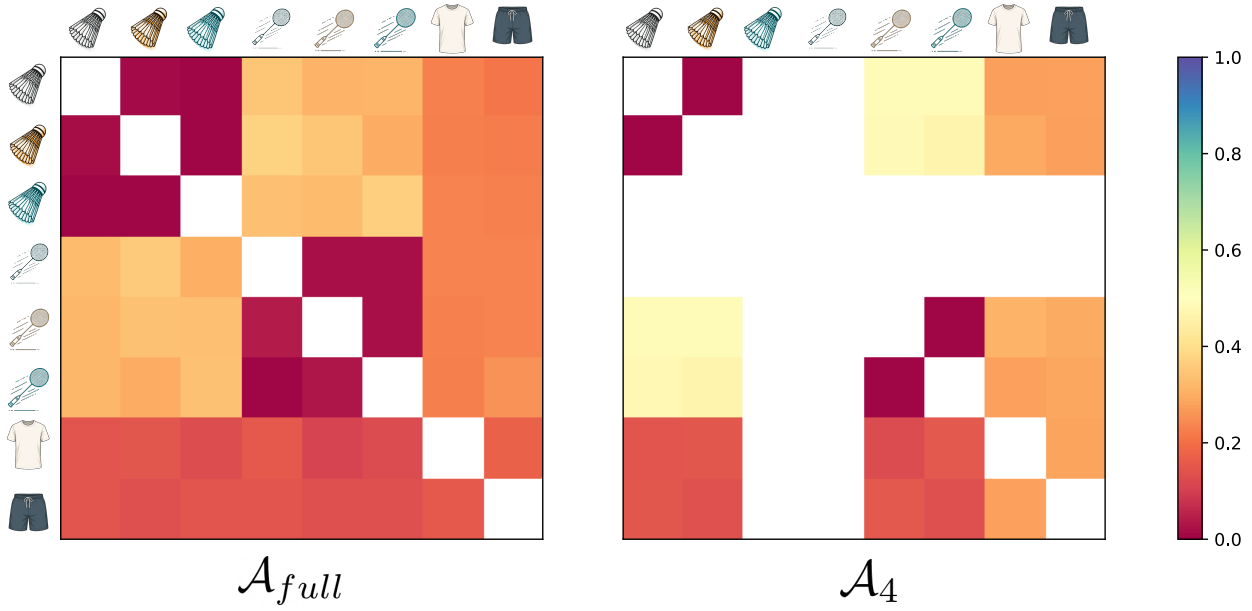


Figure 1 | $\mathbb{P}(i \in \mathcal{B} | j \in \mathcal{B})$ for all item pairs (i, j), computed on the synthetic baskets $\mathcal{D}_{full}$ on the left and $\mathcal{D}_4$ on the right. Low values correspond to cannibalization and high values to complementarity.

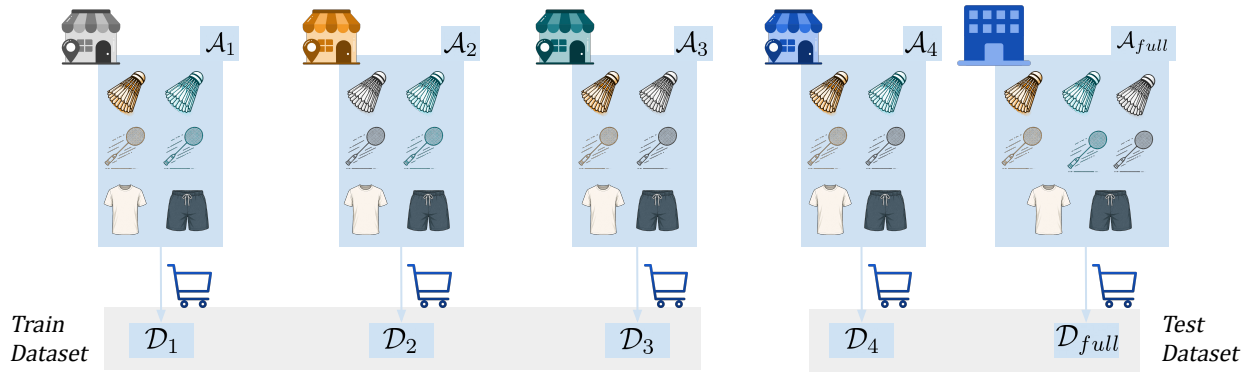


Figure 2 | Illustration of the synthetic data assortments. The full set of items $\mathcal{I}$ is in $\mathcal{A}_{full}$.

- Basket-MVL [11]: This choice model assumes the utility of a basket $\mathcal{B}$ to be the additive contributions of the marginal linear utilities of each item within the basket along with a symmetrical interaction term between those items: $U(\mathcal{B}) = \sum_{i \in \mathcal{B}} \beta \log p_i + \sum_{i,j \in \mathcal{B}} \boldsymbol{\gamma_i} \cdot \boldsymbol{\gamma_j}$. Finally, the probability for $\mathcal{B}$ to be purchased is computed with the softmax function over all possible baskets.
- Translation-Based Recommendation (TBR) [38]: It models a customer next choice as a translation from its previous choices with learned item-item interactions.
- Ours: Our model trained on $\mathcal{D}_{train}$ with negative samples drawn following Equation 7.
- Ours$_{\mathcal{I}}$: Our model trained on $\mathcal{D}_{train}$, however any item can be a negative sample as long as it is not in the basket. For this model, $\mathcal{NS} = \mathcal{I} \backslash \mathcal{B}_k$.
- Ours$_{full}$: Our model, trained on $\mathcal{D}_{full}$. This idealistic version of the model has been trained on the true underlying distribution. The two negative sampling strategies are the same for this model since $\mathcal{A} = \mathcal{I}$ in $\mathcal{D}_{full}$.

Except for Ours$_{full}$, all models are trained with basket data $\mathcal{D}_{train}$ issued from the three assortment

$\mathcal{A}_1, \mathcal{A}_2$ and $\mathcal{A}_3$, and evaluated on the test dataset $\mathcal{D}_{test}$. The differences between the models can be easily analyzed by comparing the co-purchase probabilities $\mathbb{P}(i \mid \mathcal{B} = \{j\})$ in Figure 3 with the ground truth on $\mathcal{D}_{full}$ in Figure 1. Similarly, Figure 4 presents a new product combination with assortment $\mathcal{A}_4$, which is not part of the training set. In both cases, we observe that our model, $Ours_{\mathcal{A}}$ better captures the product interactions. We also evaluate the models with two metrics, the conditional Negative Log-Likelihood NLL and the Mean Reciprocal Rank MRR computed as:

$$NLL(\mathcal{D}) = -\sum_{\mathcal{B}\in\mathcal{D}}\sum_{i\in\mathcal{B}} \log \mathbb{P}(i \mid \mathcal{B}\backslash i) \tag{8}$$

$$MRR(\mathcal{D}) = \sum_{\mathcal{B}\in\mathcal{D}}\sum_{i\in\mathcal{B}} \frac{1}{rank\left[V(i \mid \mathcal{B}\backslash i)\right]} \tag{9}$$

with $rank[.]$ being the ranking function over the available options.

The metrics, reported in Table 1, show that taking well into account the assortment while drawing the negative samples is key to capture the proper patterns. Our proposition of negative sampling $\text{Ours}_{\mathcal{A}}$ better learns the true underlying distribution, and better generalizes on new assortment variations. A first explanation can be found using the conditional probabilities illustratecad in Figures 3. Indeed, the difference is particularly visible for some pairs of complementary items, such as items $(1, 6)$ and $(2, 6)$ that have not always been sold at the same time. The models resulting from the naive negative sampling tend to underestimate the complementarity between such items (or equivalently overestimate cannibalization). Except few surprising outliers with high values, Shopper appears to worsen this effect, that we interpret as an impact of its encoding-decoding structure as well as the use of all sub-baskets during its training. Taking into account the assortment is thus beneficial to capture well the different product interactions. While this notion might not be necessary for settings such as language [9] or online retail [1], it definitely should not be overlooked for brick-and-mortar retailers.

5.2. Large scale experiments: Computation time and memory usage

Computation time and memory usage are critical when dealing with basket choice models, particularly in real-world applications where models need to process large-scale sales data efficiently. While some open-source packages offer optimized processes for data storage and discrete choice models estimation [12, 13], it is still a challenge when considering bundles of products.

For this experiment, we compare the processing time for a single epoch of training with a dataset of one million purchases, with respectively 100 and $1,000$ items considered. We compare our approach implemented in Python with TensorFlow [32] with the two other models. Shopper [1], has a C++ codebase publicly available. Additionally, we have re-implemented in Python a deterministic version of the model[3], reproducing the training procedure for a non-Bayesian neural network. We also use our own Python implementation for the basket-MVL model, whose estimation uses a composite conditional likelihood function described in [11], and the Translation-Based Recommendation [38]. The experiment is conducted on a single AMD Ryzen PRO 5955WX CPU and 250GB of RAM.

The results are reported in Table 2. First, we can observe that our approach as well as our Python implementation of Shopper scale way better in terms of dataset size, while it is competitive with the C++ implementation in terms of training speed. Our approach is also faster than the Shopper re-implementation, mainly because it does not need to consider permutations of the basket. Following these results, we discarded Shopper and Basket-MVL models for our large scale real-world dataset, with experiments described below.

[3]The code for the experiments can be found at https://github.com/artefactory/alea-carta-est.

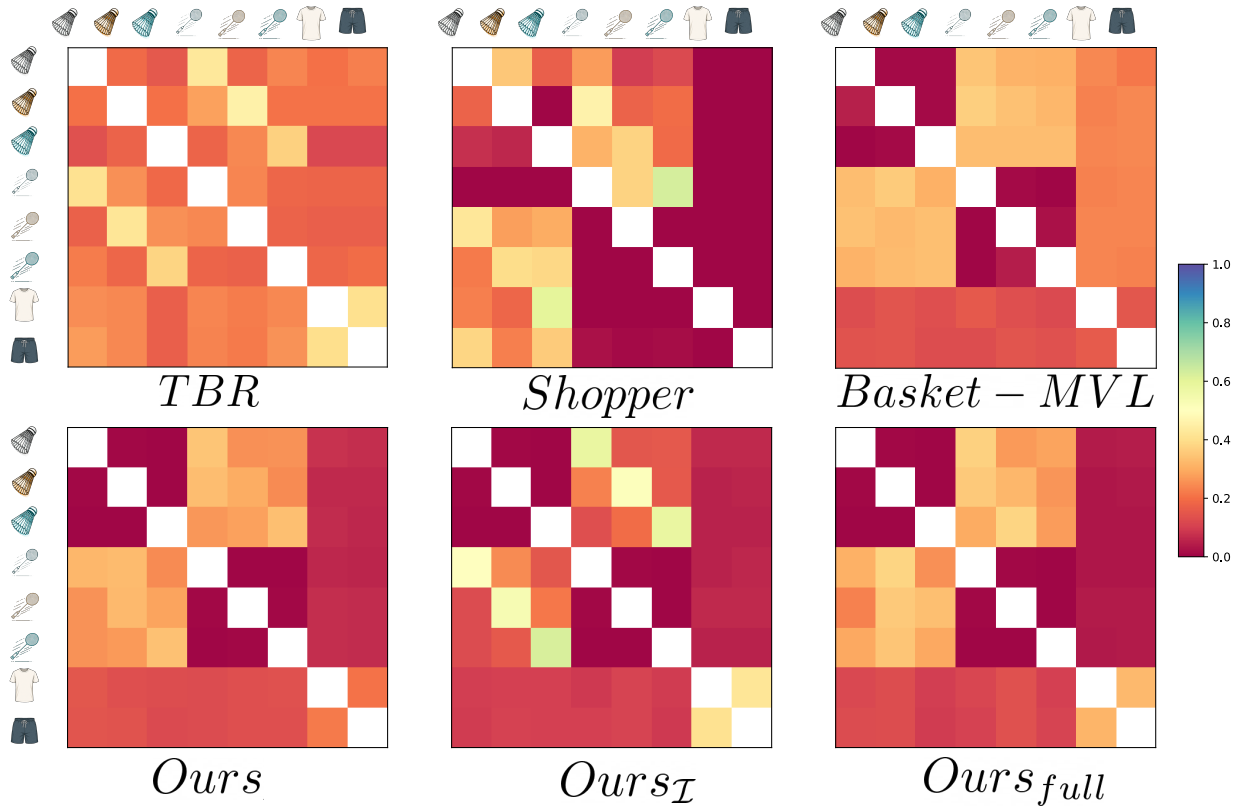


Figure 3 | Conditional probabilities $\mathbb{P}(i|\mathcal{B} = \{j\})$ for $i \neq j$ as estimated by the different models on $\mathcal{D}_{full}$. These probabilities are to be compared with the left of Figure 1.



Figure 4 | Conditional probabilities $\mathbb{P}(i|\mathcal{B} = \{j\})$ for $i \neq j$ as estimated by the different models on the test data $\mathcal{D}_4$. These probabilities are to be compared with the right of Figure 1.

Table 1 | Negative Log-Likelihood NLL (lower is better) and Mean Reciprocal Rank MRR (higher is better) for the different models on the test set.

Model	Shopper	Basket-MVL	TBR	Ours$_{full}$	Ours$_{\mathcal{I}}$	Ours
$NLL(\mathcal{D}_{full})$	1.75	1.69	1.82	**1.25**	1.43	1.26
$NLL(\mathcal{D}_4)$	1.49	1.41	1.55	0.97	1.07	**0.95**
$MRR(\mathcal{D}_{full})$	0.32	0.41	0.28	**0.42**	0.39	0.41
$MRR(\mathcal{D}_4)$	0.45	0.60	0.45	**0.62**	0.59	0.61

Table 2 | Comparison of computation time and RAM usage for different models

	100 Items		1000 Items	
	Process Time	Fits in RAM	Process Time	Fits in RAM
Shopper	100s	✓	-	×
Python-Shopper	110s	✓	620s	✓
Basket-MVL	3,000s	✓	-	×
TBR	550s	✓	1020s	✓
Ours	80 s	✓	360s	✓

5.3. Experiments on real-world retail data

We consider a dataset of purchases collected through hundreds of different supermarkets of the European retail company *Fortenova Group* during four years, from 2017 to 2021 that we describe in Table 3. Restraining the dataset to food related products that have been sold at least 1,000 times, we count more than 6,000 different products purchased. We consider a train/test temporal split to evaluate our model. The train data covers purchases from 2017 to 2020 and test data $\mathcal{D}_{2021}$ covers year 2021. In Table 4 we share some metrics results, stressing again the need for a negative sampling adapted to the assortment. The results were obtained with the same latent size $K = 128$ for all embeddings and $|\mathcal{NS}| = 20$ negative samples. We share more details of the implementation in Appendix B.

In Figure 5, we provide a T-SNE [33] projection of a subset of the learned $\boldsymbol{\gamma}_i$ embeddings of our model, colored according to product i's category. We observe that while the model is not provided with any type of information about the different products beside their price, it successfully creates clusters that correspond to products categories. It showcases that the model is able to learn a logical hierarchy of the products using the basket information.

5.4. Application: Highlighting complementary products

The main purpose of such models is to assist a category manager when planning the future assortment or promotion campaign for different stores. One basic experiment is to infer for each product a short list of the most complementary items. With such knowledge provided, the category manager can decide to make sure that complementary items are integrated within the same assortments. We propose to measure the following quantity $C(i, j) \in \mathbb{R}$ in order to rank the items j according to their complementarity with item i:

$$C(i,j) = \sum_{\mathcal{B} \in \mathcal{D}} \mathbb{P}(i \mid \mathcal{B} \cup \{j\}) + \mathbb{P}(j \mid \mathcal{B} \cup \{i\}) - \mathbb{P}(i \mid \mathcal{B} \backslash \{j\}) - \mathbb{P}(j \mid \mathcal{B} \backslash \{i\}) \tag{10}$$

Table 3 | Description of our retail dataset. Numbers are rounded for confidentiality reasons.

#Referenced Items	#Shops	#Baskets	#Assortments	Avg Basket Size
6,000+	600+	100,000,000+	10,000+	12.4

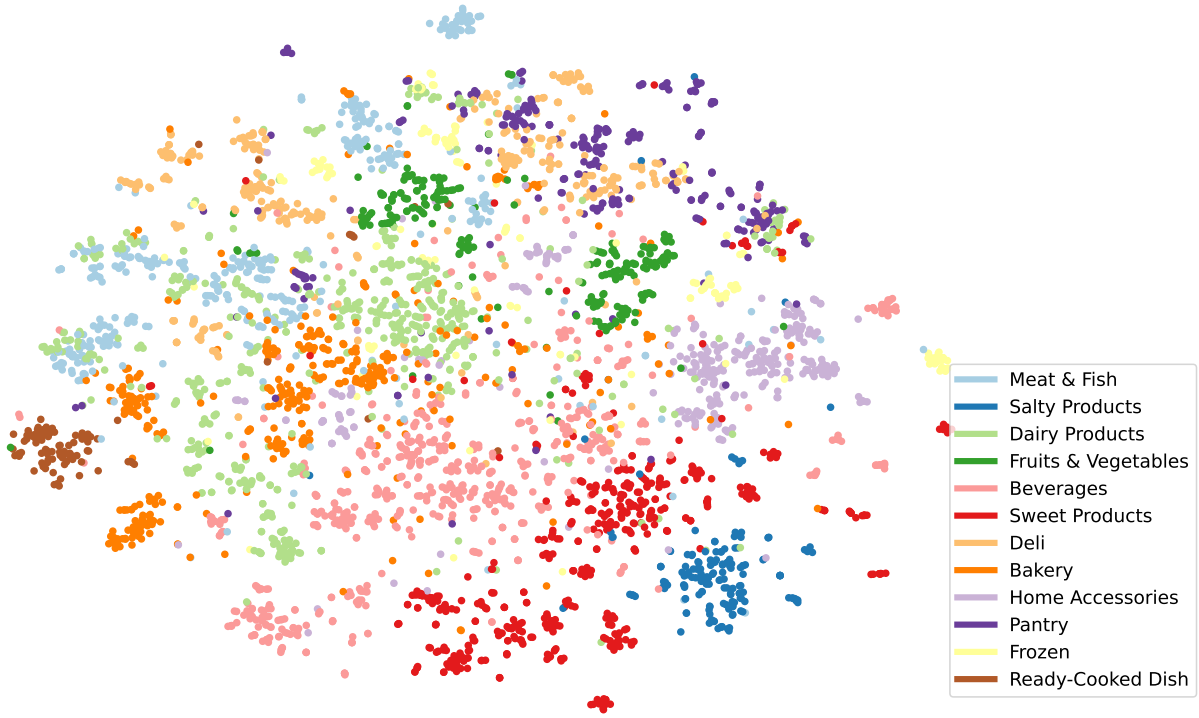


Figure 5 | T-SNE projection of the learned γ_i embeddings

which is symmetrical w.r.t i and j. The top 10 products j in terms of $C(i,j)$ values are shown in Table 5, for 4 examples of products i. In the case of tomato sauce and vanilla powder, the model outlines products that do appear highly complementary. We obtain respectively pasta, bread and pizza dough with tomato sauce and diverse ingredients to bake a cake with the vanilla powder, thus recovering meaningful ingredients together. The example of the soda bottles is also particularly interesting. Indeed, the model has understood packaging size interactions. The large soda bottle has other large bottles as complementary items, while the small soda bottles has smaller formats as top complementary products and also a single portion, to-go ice cream. It may correspond to two types of purchasing behaviors, one optimizing the price/quantity ratio with larger packages and the other one corresponding to instant consumption.

Following these promising results, we are currently conducting a survey with the company's category managers in order to confirm the relevancy of our top complementary items. They are asked to give feedback on different lists of items inferred by our model. Following these feedbacks, we plan to deploy the model as an API that would be called by several user interfaces used by category managers in their daily work. The model would regularly be trained on the sales data in order to keep up-to-date patterns.

Table 4 | Conditional Negative Log-Likelihood (lower is better) and Mean Reciprocal Rank (higher is better) on the retail test set.

Model	Ours$_{\mathcal{I}}$	Ours
$NLL(\mathcal{D}_{2021})$	3.03	**2.48**
$MRR(\mathcal{D}_{2021})$	0.015	**0.018**

Table 5 | Examples of top-10 complementary products according to our model, ranked w.r.t $C(i, j)$ (Equation 10) for the product i displayed on top. The products are referenced by their nature and brand names, replaced by a letter for anonymity. For each product displayed on top, the first-ranked complementary product is on the top left and the tenth-ranked complementary product is on the bottom right.

Soda A **0.5L**		Soda A **2L**	
Water A **0.5L**	Water B **1L**	Soda B **2L**	Soda C **2L**
Soda B **0.5L**	Water C **0.5L**	Water A **1.5L**	Sweets A **500g**
Soda C **0.5L**	Energy Drink **0.5L**	Soda D **1.5L**	Water B **1L**
Soda D **0.5L**	Strong Alcohol **0.1L**	Light Alcohol **0.5L**	Soda E **2L**
Water D **0.5L**	Ice Cream **0.16L**	Sweets B **100g**	Soda F **2L**
Tomato sauce A		Vanilla powder A	
Pasta A	Pasta B	Margarin A	Chocolate A
Bread A	Pizza Dough A	Chocolate B	Chocolate C
Pasta C	Other Sauce A	Baking Powder A	Sugar A
Toasts A	Other Sauce B	Baking Powder B	Margarin B
Bread B	Other Sauce C	Whipped Cream A	Margarin C

6. Conclusion

In this work, we introduced a machine learning model for basket choice modeling that learns embeddings of items in order to capture interactions between them. By leveraging negative sampling techniques, we addressed the computational challenges posed by large transaction datasets, ensuring efficient and scalable learning. Our experiments on synthetic data demonstrated the added value of our model and of its negative sampling method that better captures interactions between products in retail. The large-scale evaluation on more than one million transactions from a leading European retailer validated the practical applicability of our approach, with notably very promising top complementary items. Besides, our approach shows competitive predictive performances and computing scalability, in particular when compared to the models from the literature.

The results confirm the potential of our model to enhance retail decision-making, particularly in helping category managers with assortment optimization and promotion campaign planning. Beyond the direct applications by enriching user interfaces in retail, our model can be extended to other domains involving large-scale consumer choice modeling. Future work will explore the integration of additional features, such as spatial proximity between products in the store shelves and personalized customer profiles. The former will enable the identification of product interactions with respect to their distance in store, while the latter will reveal how these interactions vary across different customer segments.

References

[1] Ruiz, Francisco J.R. and Athey, Susan and Blei, David M.: SHOPPER: A Probabilistic Model of Consumer Choice with Substitutes and Complements. The Annals of Applied Statistics 14(1), 1–27, JSTOR (2020)

[2] McFadden, D.: Conditional Logit Analysis of Qualitative Choice Behavior. Frontiers in Econometrics, 105–142, Academic press (1974)

[3] Wang, S. and Mo, B. and Zhao, J.: Deep neural networks for choice analysis: Architecture design with alternative-specific utility functions. Transportation Research Part C: Emerging Technologies 112, 234–251, Elsevier (2020)

[4] McFadden, Daniel et al.: Modelling the choice of residential location. Institute of Transportation Studies, University of California California (1978)

[5] Han, Y. and Pereira, F. C. and Ben-Akiva, M. and Zegras, C.: A neural-embedded discrete choice model: Learning taste representation with strengthened interpretability. Transportation Research Part B: Methodological 163, 166–186 (2022)

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

[7] Sobolewski, Maciej and Kopczewski, Tomasz: Estimating demand for fixed-line telecommunication bundles. Telecommunications Policy 41(4), 227–241, Elsevier (2017)

[8] Arkoudi, Ioanna and Krueger, Rico and Azevedo, Carlos Lima and Pereira, Francisco C.: Combining discrete choice models and neural networks through embeddings: Formulation, interpretability and performance. Transportation research part B: methodological 175, 102783, Elsevier (2023)

[9] Mikolov, Tomas and Sutskever, Ilya and Chen, Kai and Corrado, Greg S and Dean, Jeff: Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems 26 (2013)

[10] Daniel McFadden and Kenneth Train: Mixed MNL Models for Discrete Response. Journal of Applied Econometrics 15(5), 99–110, Wiley (2000)

[11] Vincenzina Caputo and Jayson L. Lusk: The Basket-Based Choice Experiment: A Method for Food Demand Policy Analysis. Food Policy 109, 102252 (2022)

[12] Auriau, Vincent and Aouad, Ali and Désir, Antoine and Malherbe, Emmanuel: Choice-learn: large-scale choice modeling for operational contexts through the lens of machine learning. Journal of Open Source Software 9(101), 6899, The Open Journal (2024)

[13] Tianyu Du and Ayush Kanodia and Susan Athey: Torch-Choice: A PyTorch Package for Large-Scale Choice Modelling with Python. arXiv preprint arXiv:2304.01906 (2023)

[14] Balestriero, Randall and Ibrahim, Mark and Sobal, Vlad and Morcos, Ari and Shekhar, Shashank and Goldstein, Tom and Bordes, Florian and Bardes, Adrien and Mialon, Gregoire and Tian, Yuandong and others: A cookbook of self-supervised learning. arXiv preprint arXiv:2304.12210 (2023)

[15] Gong, Yunchao and Ke, Qifa and Isard, Michael and Lazebnik, Svetlana: A multi-view embedding space for modeling internet images, tags, and their semantics. International journal of computer vision 106, 210–233, Springer (2014)

[16] Liu, Cai-zhi and Sheng, Yan-xiu and Wei, Zhi-qiang and Yang, Yong-Quan: Research of text classification based on improved TF-IDF algorithm. IEEE international conference of intelligent robotic and control engineering (IRCE), 218–222 (2018)

[17] Yang, Zhen and Ding, Ming and Huang, Tinglin and Cen, Yukuo and Song, Junshuai and Xu, Bin and Dong, Yuxiao and Tang, Jie: Does negative sampling matter? a review with insights into its theory and applications. IEEE Transactions on Pattern Analysis and Machine Intelligence (2024)

[18] Pilehvar, Mohammad Taher and Camacho-Collados, Jose: Embeddings in natural language processing: Theory and advances in vector representations of meaning. Morgan & Claypool Publishers (2020)

[19] LeCun, Yann and Bengio, Yoshua and Hinton, Geoffrey: Deep learning. Nature 521, 436–444, Nature Publishing Group UK London (2015)

[20] Guo, Cheng and Berkhahn, Felix: Entity embeddings of categorical variables.arXiv preprint arXiv:1604.06737 (2016)

[21] Seger, Cedric: An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing (2018)

[22] Rodríguez, Pau and Bautista, Miguel A and Gonzalez, Jordi and Escalera, Sergio: Beyond one-hot encoding: Lower dimensional target embedding. Image and Vision Computing 75, Elsevier (2018)

[23] Akata, Zeynep and Perronnin, Florent and Harchaoui, Zaid and Schmid, Cordelia: Label-embedding for image classification. IEEE transactions on pattern analysis and machine intelligence 38(7), 1425–1438 (2015)

[24] Xu, Mengjia: Understanding graph embedding methods and their applications. SIAM Review 63(4), 825–853 (2021)

[25] Kim, Myung Jun and Grinsztajn, Léo and Varoquaux, Gaël: CARTE: pretraining and transfer for tabular learning. arXiv preprint arXiv:2402.16785 (2024)

[26] Koren, Yehuda and Rendle, Steffen and Bell, Robert: Advances in collaborative filtering. Recommender systems handbook, 91–142, Springer (2021)

[27] Li, Shiwei and Guo, Huifeng and Tang, Xing and Tang, Ruiming and Hou, Lu and Li, Ruixuan and Zhang, Rui: mbedding compression in recommender systems: A survey. ACM Computing Surveys 56(5), 1–21, ACM New York (2024)

[28] ACM Transactions on Management Information Systems (TMIS): The netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS) 6(4), 1–19, ACM New York (2015)

[29] Molenberghs, Geert and Verbeke, Geert et al.: Models for discrete longitudinal data. Springer 22 (2005)

[30] Gutmann, Michael U and Hyvärinen, Aapo: Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. The journal of machine learning research 13(1), 307–361, JMLR (2012)

[31] Salvadé, Nicolas and Hillel, Tim: RUMBoost: Gradient boosted random utility models. Transportation Research Part C: Emerging Technologies 170, 104897, Elsevier (2025)

[32] Abadi, Martín and Agarwal, Ashish and Barham, Paul and Brevdo, Eugene and Chen, Zhifeng and Citro, Craig and Corrado, Greg S and Davis, Andy and Dean, Jeffrey and Devin, Matthieu et al.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 (2016)

[33] Van der Maaten, Laurens and Hinton, Geoffrey: Visualizing data using t-SNE. Journal of machine learning research 9(11) (2008)

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

[35] Sifringer, Brian and Lurkin, Virginie and Alahi, Alexandre: Enhancing discrete choice models with neural networks. 18th Swiss Transport Research Conference, 1–3 (2018)

[36] Pereira, Francisco C.: Rethinking travel behavior modeling representations through embeddings. arXiv preprint arXiv:1909.00154 (2019)

[37] Ko, Joohwan and Li, Andrew A.: Modeling Choice via Self-Attention. arXiv preprint arXiv:2311.07607 (2023)

[38] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based Recommendation. In Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17). Association for Computing Machinery, New York, NY, USA, 161–169. https://doi.org/10.1145/3109859.3109882

A. Synthetic Data Generation

We describe the iterative algorithm used for the experiments in Section 5.1 to obtain synthetic data. From an assortment $\mathcal{A}$, we sample the purchased baskets as follows:

1. Selection of the first item We consider that the following sets can be chosen with the same probability:

- $\{1, 2, 3\} \cap \mathcal{A}$ the ensemble of available shuttlecocks
- $\{4, 5, 6\} \cap \mathcal{A}$ the ensemble of available rackets
- $\{7\} \cap \mathcal{A}$ the ensemble of available shorts
- $\{8\} \cap \mathcal{A}$ the ensemble of available t-shirts

One item is then chosen among the selected set, all items also having the same probability.

2. Basket Completion Once the first item i is chosen, one item by remaining nest can be added to the basket. Probabilities $\mathbb{P}(i|j \in \mathcal{B})$ equal to 0.7 for nest with complementary items and 0.15 for neutral items. Once a nest is selected, one items among the nest is finally selected, all having the same probability.

3. Adding some random noise With a probability $P = 0.15$ one of the items of the assortment, not already in the basket, may also be added.

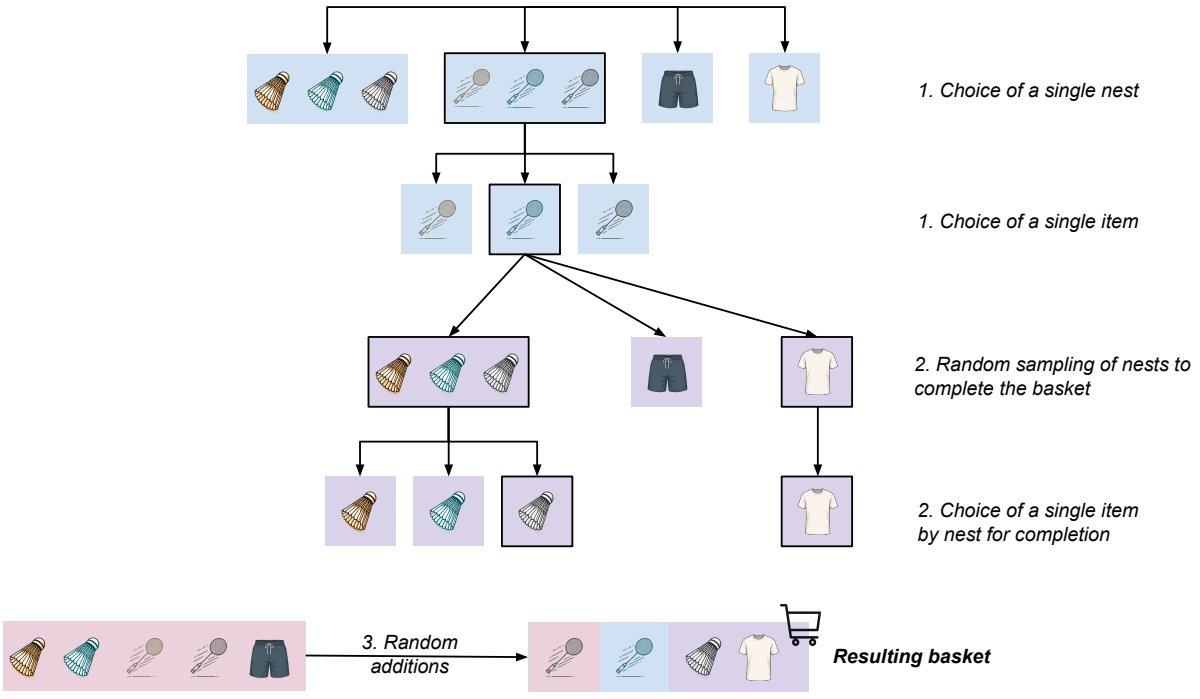


Figure 6 | Successive steps for synthetic data sampling from an assortment. In this example, the racket nest is selected during the first step, with the green racket finally being selected. During the step 2, the grey shuttlecock and the t-shirt are added. Finally a second racket is added to the basket during step 3.

B. Experiments on real-world retail data

For the experiments on real-world data, we used the following hyperparameters:

- α, β latent size: 12
- γ latent size: 128
- Number of negative samples: $|\mathcal{NS}| = 24$
- Optimizer: Adamax with a learning rate of 0.001
- Batch size: 256
- Number of epochs: 20