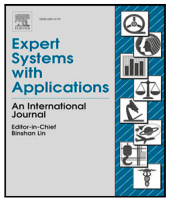
Expert Systems With Applications 212 (2023) 118795

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Next-basket prediction in a high-dimensional setting using gated recurrent units

Luuk van Maasakkers [a,*], Dennis Fok [b], Bas Donkers [c]

[a] Econometric Institute & Department of Business Economics, Erasmus School of Economics, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, Netherlands

[b] Econometric Institute, Erasmus School of Economics, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, Netherlands

[c] Department of Business Economics, Erasmus School of Economics, Erasmus University Rotterdam, P.O. Box 1738, 3000 DR Rotterdam, Netherlands

ARTICLE INFO

Keywords:
Next-basket prediction
Gated recurrent units
Neural networks
Machine learning

ABSTRACT

Accurately predicting the next shopping basket of a customer is important for retailers, as it offers an opportunity to serve customers with personalized product recommendations or shopping lists. The goal of next-basket prediction is to predict a coherent *set* of products that the customer will buy next, rather than just a *single* product. However, if the assortment of the retailer contains thousands of products, the number of possible baskets becomes extremely large and most standard choice models can no longer be applied. Therefore, we propose the use of a gated recurrent unit (GRU) network for next-basket prediction in this study, which is easily scalable to large assortments. Our proposed model is able to capture dynamic customer taste, recurrency in purchase behavior and frequent product co-occurrences in shopping baskets. Moreover, it allows for the inclusion of additional covariates. Using two real-life datasets, we demonstrate that our model is able to outperform both naive benchmarks and a state-of-the-art next-basket prediction model on several performance measures. We also illustrate that the model learns meaningful patterns about the retailer's assortment structure.

1. Introduction

Nowadays, retailers can easily record the purchase behavior of their customers by means of customer accounts and loyalty cards. Accurately predicting the future purchase behavior of their customers still remains an important challenge for retailers, though. Based on such predictions, customers can be served with individual-specific product recommendations or personalized promotions. By doing so, a retailer can increase customer satisfaction and stimulate customer retention, resulting in higher future sales (Montgomery & Smith, 2009; Rust & Chung, 2006; Zhang & Wedel, 2009). However, the underlying factors influencing a customer's decision process are numerous and can be complex to capture.

In past literature, many models have been proposed for the prediction of customer purchase behavior and for personalized recommendations of products. Most of these models rank single products based on their estimated attractiveness to customers. However, in several real-life situations (e.g. grocery shopping), customers purchase a set of multiple products (a *basket*) at the same time. In such cases, it is useful to recommend a set of products to a customer that is not only in line with the customer's preferences, but also takes into account co-occurrence patterns of products within the basket. To capture such patterns, it is important to jointly model the entire set of products purchased at the next shopping trip. This, however, comes with the problem that the set of possible baskets grows exponentially with the size of the retailer's assortments. Nowadays, most (online) retailers sell thousands of products and for them, many standard prediction techniques are infeasible to apply. In this study, we propose a novel, scalable next-basket prediction model based on Gated Recurrent Units (GRU), a specific type of recurrent neural network. We demonstrate that our model can make accurate basket predictions for assortments containing thousands of products.

In the past decade, several models have been proposed for next-basket prediction based on personalized Markov chains (Rendle, Freudenthaler, & Schmidt-Thieme, 2010), hierarchical representation models (Wang et al., 2015), neural networks (Bai et al., 2018; Le, Lauw, & Fang, 2019; Yu, Liu, Wu, Wang, & Tan, 2016) and temporal annotated recurrent sequences (Guidotti, Rossetti, Pappalardo, Giannotti, & Pedreschi, 2017). Most of these models consider only one or two of the following drivers of basket purchases: customer baseline taste (relatively stable, potentially slowly changing over time), sequential basket patterns (e.g. driven by inventory depletion) and co-occurrence patterns within baskets. Our GRU-based approach captures

* Corresponding author.
E-mail addresses: vanmaasakkers@ese.eur.nl (L. van Maasakkers), dfok@ese.eur.nl (D. Fok), donkers@ese.eur.nl (B. Donkers).

https://doi.org/10.1016/j.eswa.2022.118795
Received 9 May 2022; Received in revised form 5 September 2022; Accepted 5 September 2022
Available online 10 September 2022

all these drivers to make accurate predictions of the next shopping basket. Moreover, our approach also allows for inclusion of (basket-specific) covariates such as seasonality, price or promotion variables and generates corresponding insights. The resulting model structure also enables identification of similarities between various products, providing face validity to the model structure in addition to a strong predictive performance.

Below, we will first further discuss the problem context and we review the literature. In the next section, we will provide a detailed description of the proposed model. Afterwards, we apply the model on two real-life data sets. In the results section we report our findings, where we look at predictive performance and illustrate the behavioral features that are captured by the model. We end with some concluding remarks.

2. Context and literature review

In this section, we describe three streams of related literature to which our research fits. First, we consider the recent stream of research that has developed models for high-dimensional forecasting in marketing. Secondly, we dive deeper into models that have been developed in recent years for the specific task of next-basket prediction. Finally, we discuss the theoretical background from the marketing literature for the behavioral patterns we aim to capture with our model.

2.1. High-dimensional forecasting

Most classical parametric forecasting techniques are no longer feasible for purchase predictions when there are thousands of alternatives to choose from. The rise of online stores selling thousands of products and the recent growth in the usage of these web shops by customers has not only resulted in a growth in data availability, but also an increased interest in marketing models that can handle such large-scale assortments. This scenario requires completely new types of models. Machine learning techniques seem very promising. Such methods usually have a lot of parameters and are more complex than the traditional models. Computational advances in recent decades have made it possible to estimate parameters of such complex models. Because of these recent developments, the topic of large-scale assortments is relatively new in marketing modeling and forecasting.

Several models have been proposed in recent years to capture purchase behavior in the scenario of large-scale assortments. Jacobs, Donkers, and Fok (2016) and Jacobs, Fok, and Donkers (2021) propose a scalable purchase prediction model based on latent Dirichlet allocation. It is shown that this model outperforms a collaborative filter (Sarwar, Karypis, Konstan, & Riedl, 2001) and a discrete choice model in terms of prediction performance. Although the model allows for customer heterogeneity, it does not capture effects of price and promotions. Gabel and Timoshenko (2021) proposed a neural network that takes purchase history and coupon discounts as inputs to predict future purchase behavior. By doing so, the trained network could be used to optimize the personalization of coupons. Our research contributes to this line of literature by proposing a next-basket prediction model that is able to deal with large-scale assortments and also allows for inclusion of basket-specific covariates such as price, promotions and seasonality.

2.2. Next-basket prediction

Next-basket prediction is a relatively new research area: most proposed models have been developed over the past decade. Studies in the field of next-basket prediction aim to predict the basket of a certain user at time t based on the purchased baskets at time $t-1$, $t-2$, etc. These time points are generally not absolute, but relative to a customer's shopping activities (e.g. their first basket, second basket, etc.).

Rendle et al. (2010) propose a factorized personalized Markov chain (FPMC) for next-basket prediction. This model combines matrix factorization (MF, to capture general taste of users) and Markov chains (MC, to model purchase dynamics) into a Markov chain with user-specific transitions. The state in FPMC codes actual basket realizations, such that the state space has size 2^N, with N the number of products in the assortment. This makes it infeasible to model long chains, therefore only first-order Markov chains are considered. Transition matrices are individual-specific, but matrix factorization is used to propagate information among similar users, similar items and similar transitions. The researchers show that the proposed model outperforms standard MF and MC models on an online drug store dataset with over 7000 products. Their focus lies only on recommending products to a customer that where not bought earlier by that customer, although their model can also be used for next-basket prediction.

The FPMC model was extended to the hierarchical representation model (HRM) by Wang et al. (2015). In this model, both items and users are represented by latent vectors of equal dimensions, describing characteristics of the items and users, with non-linearities added through various pooling operations. Different specifications of these non-linearities match the basic Markov chain model, matrix factorization model or the FPMC model, enabling the HRM model to outperform all.

Yu et al. (2016) describe a dynamic recurrent model (DREAM) based on a recurrent neural network. Similar to the HRM, items are described by latent vectors, which are pooled to obtain a basket latent vector. Next, these basket-representing latent vectors of a user's purchase history are mapped to score vectors through non-linear operations in the recurrent neural network, where each element of the score vector represents one item. A high score indicates that the user is more likely to purchase the corresponding item. The network contains one hidden layer that can be interpreted as a dynamic representation of the user, as it is based on all earlier purchased baskets and gets updated when a new basket is added. In addition, the recurrency of the network takes the sequential behavior of users into account. The researchers show that DREAM outperforms other models such as FPMC and HRM.

Finally, the Temporal Annotated Recurring Sequences (TARS) model by Guidotti et al. (2017) takes a different approach. In contrast to the previously discussed studies, Guidotti et al. (2017) aim to construct a model that is readable and interpretable by humans. The model is mainly based on item co-occurrence and recurring sequences. A sequence, as defined by the researchers, consists of two sets of items, where the first set is contained in a basket purchased earlier than the basket containing the second set. A recurring sequence is defined as a sequence that occurs at least a certain number of times in the purchase history. The temporal annotated recurring sequences can be used to make predictions of new baskets, by taking into account all recurring sequences that are potentially active at the next purchase time. Predictions are personalized and user-specific, meaning that the predictions for a certain customer depend on a model that is built based on only the purchase history of that customer. This is in line with the user-centric vision for data prediction, as noted by the researchers. A drawback of the model is that customer histories need to be relatively long in order to produce accurate prediction. For example, Guidotti et al. (2017) consider a subset of customers of the Chinese retailer Ta-Feng with at least 10 basket purchases, removing 93% of the original customers. This makes the model difficult to apply in situations where limited data is available per customer. The model we propose does not have such a cold-start problem and predicts accurately for all customers with at least one prior basket purchase. TARS is shown to outperform several benchmarks in next basket prediction, including the three earlier discussed methods. In turn, we will demonstrate that our GRU-based approach outperforms TARS, for both short and long basket sequences.

Our approach shares features with the existing methods, but also improves upon each of them. We do not have the restrictions of the FPMC, that can only be estimated for chains of two consecutive baskets, or TARS, that requires relatively long basket sequences in order to be

estimated accurately. In comparison with DREAM, the GRU dynamics allow us to model the relation between subsequent baskets and product co-occurrence patterns in a more flexible way. DREAM aggregates all latent product representations by averaging or taking the element-wise maximum, before entering the recurrent layers. Instead, we do not pool products before entering the recurrent layers, thereby allowing for more flexible interactions. We also extend these existing methods by allowing for the inclusion of additional price, promotion and/or seasonality variables.

2.3. Model components

We distinguish three drivers of purchase behavior that we aim to model: *baseline customer taste* (assumed to be relatively stable over time), *sequential basket patterns* and *item co-occurrence or co-purchase patterns*. Single-item prediction models usually only capture the first and (sometimes) the second driver. However, for next-basket prediction, capturing the third dimension is also important, as it enables a model to recommend a coherent set of products to a customer. In this subsection, we discuss each of these drivers one-by-one and provide the theoretical background based on the marketing literature.

First, our model should be able to capture the baseline taste of each customer. Previous studies have found that most customers show habit persistence in their purchase decisions (Erdem, 1996; Kannan & Sanchez, 1994). Habit persistence, also known as purchase reinforcement, can be defined as the reinforcement of tastes or preferences by past behavior over time (Erdem, 1996). From a modeling perspective, habit persistence implies that past product purchases positively affect the purchase likelihood for the same or similar products in the future, through consumer-preference reinforcement. Consequently, customers often repeatedly purchase specific types of products (e.g. a specific brand, vegan products, eco-friendly products, etc.). Bronnenberg, Dubé, and Gentzkow (2012) found evidence in American home scanner data that such customer preferences, once formed, are highly persistent over time. Preferences of households that migrate to a different state are compared to preferences of non-migrants in the state they move to. A priori, migrants have very different brand preferences because of different past experiences in their state of origin. Although their preferences slowly shift towards the average of their new state, migrants still have significantly different preferences 50 years after migration. We aim to capture these persistent preferences in the latent state of our model, which is a customer-specific function of prior basket purchases. Recent product purchases can reinforce the preference for that brand or product in the latent state and, thereby, positively affect the purchase probability of that product in the future.

Despite the existence of habit persistence, consumers often also exhibit a certain degree of variety seeking in their purchase behavior (Chintagunta, 1999; McAlister & Pessemier, 1982). McAlister and Pessemier (1982) make a distinction between direct varied behavior and derived varied behavior. Direct varied behavior results from the inherent satisfaction of change and novelty. Such satisfaction can, for example, be caused by a desire for unfamiliar products, a desire to alternate between familiar products, a need for information or a need for group affiliation. On the other hand, derived varied behavior is caused by other forces, not related to a preference for change itself (e.g. because a household consists of different individuals with different preferences, or because of changes in the assortment or marketing mix of a retailer). McAlister (1982) proposed that each consumption history can be represented as an accumulation of product attributes, a so-called *attribute inventory*. With each consumed product, there is a discrete jump in the household's inventory of the attributes that constitute the product. As inventories deplete, preferences for products with the corresponding attribute increase. McAlister (1982) found empirical evidence for this inventory accumulation and decumulation process. To be able to capture such short-term sequential patterns, we also use the most recent basket as input in each prediction. Customers might decide not to purchase a certain product because they have already accumulated a large inventory of the product's attributes through recent product purchases that affect the latent state, resulting in variety seeking. Similarly, latent state information can be 'forgotten' through a so-called 'forget gate', thereby allowing the attribute inventory to also decumulate resulting in customers purchasing similar products again after enough time has passed. Habit persistence and variety seeking might seem to contradict each other, but they can go hand in hand. Habit persistence can cause a consumer to always choose from a limited subset of products, whereas variety seeking can explain alternating purchase behavior within this subset. The extent to which consumers are habit persistent or variety seeking can be individual-specific and estimated from data.

Whereas customers exhibit variety-seeking behavior over time, this behavior is also exhibited within a basket. Instead of purchasing many products that all match the need for one single attribute, customers purchase sets of products that meet the needs for multiple attributes. In situations where groups of items are chosen, the selection of one product is dependent on the selection of others (McAlister, 1979). Often, customers purchase sets of products that complement each other, e.g. to prepare a meal with several ingredients. A separate stream of literature on market basket analysis is focused on identifying such co-occurrence patterns of products in baskets (Agrawal, Imieliński, & Swami, 1993; Chen, Tang, Shen, & Hu, 2005). Bel, Fok, and Paap (2018) and Russell and Petersen (2000) used multivariate logit models to describe correlated binary purchase decisions and identified clear demand dependencies across categories during shopping trips. In order to capture such co-occurrence patterns in our model, all products from the same basket enter the model simultaneously, allowing for interactions between them. Because our model summarizes basket information in a lower-dimensional latent state (compared to the dimension of the basket itself) before making a prediction, it is forced to learn patterns that simultaneously affect multiple products. This ensures that the model learns within-basket dependencies across the products.

The fact that a GRU-based model can naturally capture each of the above three drivers of purchase behavior, makes it a very suitable model for next basket prediction. We focus on these three drivers as their effects can be learned from only transactional data. However, if additional information (e.g. regarding prices, promotions or purchase timing) is available, additional drivers could be considered, such as marketing mix and seasonality effects. Vilcassim and Jain (1991) find that such variables often have significant effects on brand switching, indicating the importance of including such variables in the model, when available. We propose a way to include such variables in the GRU-based model.

3. GRU-based next-basket prediction

In this section, we first mathematically formulate the considered problem of next-basket prediction. Next, we describe our proposed GRU-based model. Afterwards, we discuss its estimation and we conclude with a discussion of hyperparameter tuning and regularization.

3.1. Problem formulation

Consider a large set of C customers, each purchasing a series of baskets over time. Here, a basket is defined as an unordered set of products from the retailer's assortment, simultaneously bought by a customer at a certain point in time. At each moment of purchase, a customer selects a number of products from an assortment of N products. Together, the chosen products form the basket purchased at that moment in time. The basket of customer c purchased at time t is represented by an N-dimensional dummy vector $\mathbf{b}_t^c$, where the nth element b_{tn}^c is 1 if the nth product in the assortment is purchased at time t and 0 otherwise. In case the same product is purchased multiple times, it is still encoded as a 1 in our basket representation. Including

quantities in the basket vector is possible, but could be arbitrary given the different volumes and weights of products. Moreover, for marketing purposes it is often most interesting to know whether or not a customer is interested in a product, rather than the quantity a customer wants to purchase. Hence, our goal is to predict the basket of customer c at time t, $\mathbf{b}_t^c$, based on his previously purchased baskets $\mathbf{b}_{t-1}^c$, $\mathbf{b}_{t-2}^c$, etc. Note that time is relative in this notation, i.e. only the order of basket matters and not the absolute (inter)purchase times. The length of a customer c's basket sequence is denoted by T_c and can be different across customers. A summary table with explanations of all notation used in this section is provided in Appendix A.

We mainly focus on the scenario where N, the number of products in the retailer's assortment, is large (> 1000). In these cases, the observed basket vectors of customers are very sparse, meaning that only a small number of products is purchased compared to the total number of products in the assortment. Most traditional forecasting techniques become unreliable or computationally infeasible in this case.

3.2. Gated recurrent unit dynamics for next-basket prediction

In this section we introduce our next-basket prediction model that builds on gated recurrent units (GRUs) (Chung, Gulcehre, Cho, & Bengio, 2014). Fig. 1 provides an overview of the model, that we will describe in further detail below. Note that the customer superscript c is left out for convenience, but throughout this section we will still consider baskets specific to an individual.

The GRU is a specific type of recurrent neural network. Through feedback loops, information in recurrent neural networks is transferred from one time step to the next. This makes it possible to predict future baskets of customers, while taking the complete sequence of previous purchases into account. Recurrent neural networks have been used in various forms. In the standard recurrent neural network, all layers are fully-connected, meaning that the output of each layer is the result of a weighted sum of the previous layer outputs, passed through a (usually non-linear) activation function. These fully-connected recurrent neural networks often have difficulties in learning long-term dependencies, due to vanishing loss-function gradients (Bengio, Simard, Frasconi, et al., 1994). To solve this problem, Hochreiter and Schmidhuber (1997) introduced the *Long Short-Term Memory* (LSTM) network, that replaced the standard recurrent layer by a block of element-wise update operations (so-called *gates*). By using element-wise updates, loss function gradients do not vanish as fast as in fully connected neural networks and the network is able to learn long-term dependencies. Although the LSTM has shown impressive performance in natural language processing and sequential prediction tasks, its structure is very complex and difficult to interpret.

Chung et al. (2014) showed that a gated network with a simpler structure could reach predictive performance similar to the performance of an LSTM, with fewer parameters and less computation time. This network is called the gated recurrent unit (GRU). Whereas the LSTM contains two hidden states, that are updated through three gates, the GRU only contains one hidden state that is updated through two gates. This makes the model computationally less expensive and easier to interpret, without losing predictive performance. Since then, GRUs have been successfully applied in all kinds of contexts, ranging from natural language processing to marketing (Koehn, Lessmann, & Schaal, 2020). In preliminary experiments, we also considered the use of LSTM, but it did not outperform the GRU, while taking longer to train.

The hidden state in the GRU network is denoted by the d-dimensional vector $\mathbf{a}_t$, where d is a hyperparameter to be specified by the user in advance. In practice, d is chosen much smaller than the assortment size N, in order to summarize basket contents efficiently and reduce computation time. The hidden state is initialized as $\mathbf{a}_1$ (usually a vector of zeros) and updated along the horizontal line in Fig. 1, each time a new basket of the same customer is passed through

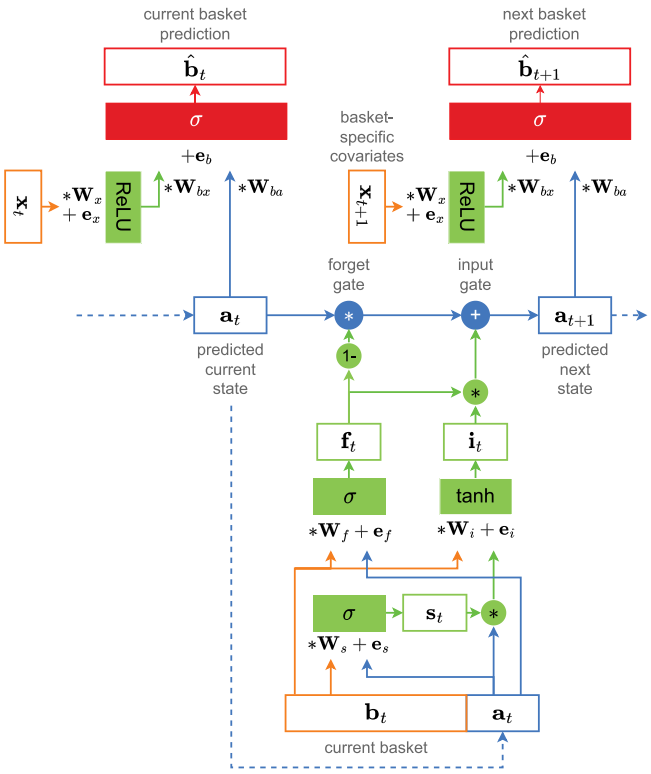


Fig. 1. Outline of the GRU-based next basket prediction model. White rectangles denote vectors, solid objects denote element-wise vector operations. σ, tanh, and ReLU respectively denote the sigmoid, hyperbolic tangent and rectifier activation function.

the network. The contents of the hidden state could be regarded as a representation of dynamic user preferences, that may change over time.

The GRU models dynamics in the latent preference state through two stages referred to as *gates*. Similar to a physical gate, they determine which information is allowed to flow through to the next time step and which information is not. The first gate, referred to as the *forget gate*, determines which information from the previous hidden state is forgotten, and which information is preserved. The second state, referred to as the *input gate*, supplements the old information with new information from the newly purchased basket. Formally, the hidden state update becomes

$$\mathbf{a}_{t+1} = (1 - \mathbf{f}_t) \odot \mathbf{a}_t + \mathbf{f}_t \odot \mathbf{i}_t, \tag{1}$$

where $\mathbf{a}_t$ is the hidden state at time t and $\mathbf{f}_t$ is the forget vector with values between 0 and 1, quantifying the extent to which information needs to be forgotten for each element of the latent state. $\mathbf{i}_t$ is a vector with values between -1 and 1, containing new information to be added to the hidden state. Finally, $\odot$ denotes the element-wise multiplication operator. Because of the element-wise updates, the hidden state can simultaneously contain persistent as well as dynamic customer preferences.

The degree of forgetting and hence also the degree to which new information is added to the latent state information depends on the most recently purchased basket and the previous hidden state. The forget vector $\mathbf{f}_t$ is formally defined as

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \cdot [\mathbf{a}_t, \mathbf{b}_t] + \mathbf{e}_f), \tag{2}$$

where $\mathbf{W}_f$ and $\mathbf{e}_f$ are model parameters and σ denotes the element-wise sigmoid function. From a regression perspective, $\mathbf{W}_f$ can be regarded as a matrix of coefficients and $\mathbf{e}_f$ as a vector of intercepts. The sigmoid function, given by

$$\sigma(x) = \frac{1}{1 + \exp(-x)}, \tag{3}$$

ensures that all outputs are between 0 and 1, and introduces non-linearity in the model.

The vector with new information, $\mathbf{i}_t$, also depends on the current observation and the current latent state as follows:

$$\mathbf{i}_t = \tanh(\mathbf{W}_i \cdot [\mathbf{s}_t \odot \mathbf{a}_t, \mathbf{b}_t] + \mathbf{e}_i), \tag{4}$$

where $\mathbf{W}_i$ and $\mathbf{e}_i$ are model parameters and tanh denotes the element-wise hyperbolic tangent function. The hyperbolic tangent function is a non-linear function that provides outputs between -1 and 1. The vector $\mathbf{s}_t$ is a scaling vector which is also allowed to depend on the previous state and previous observation. It determines the extent to which the previous state influences the next state update. The GRU defines the scaling vector (also called *reset gate*) as

$$\mathbf{s}_t = \sigma(\mathbf{W}_s \cdot [\mathbf{a}_t, \mathbf{b}_t] + \mathbf{e}_s), \tag{5}$$

with model parameters $\mathbf{W}_s$ and $\mathbf{e}_s$. This scaling vector contains values between 0 and 1, that reduce (or even ignore) information from the previous hidden state such that it becomes less relevant for the current update. Hence, this additional scaling layer is similar to the forget gate, but the forget gate permanently shrinks preference state elements, whereas the scaling layer only shrinks them temporarily in order to compute the current information update. If the scaling vector equals zero, the GRU gates treat the current basket as if it were the first in the sequence, without taking prior information into account. By adding this additional layer, the GRU can also capture interactions between the previous state and observation when updating the state.

Strictly speaking, the gated recurrent unit comprises the latent state updates and resulting dynamics as described by the update equations stated above. To obtain basket predictions, we add an output layer that maps the d-dimensional hidden state to an N-dimensional output vector, containing purchase probabilities for all products in the assortment. The fact that $d \ll N$ forces the model to learn patterns that simultaneously affect multiple products. This ensures that the model learns within-basket dependencies across the products. Formally, the basket predictions are computed as

$$\hat{\mathbf{b}}_t = \sigma(\mathbf{W}_{ba} \cdot \mathbf{a}_t + \mathbf{e}_b), \tag{6}$$

where $\mathbf{W}_{ba}$ and $\mathbf{e}_b$ are again model parameters. The sigmoid function ensures that all output probabilities are between 0 and 1. Note that these probabilities do not need to sum to 1, as multiple products can be purchased simultaneously.

3.3. Including additional covariates

Basket predictions follow from the latent state as formulated in (6). In this formulation, no additional covariates are considered yet. In many cases such covariates may be available. Price, promotion and seasonality variables may contain important information about a customer's purchase decision. By adding an additional branch before the output layer, we also allow for inclusion of such covariates in the model. The covariates can hence adjust the history-based predictions upwards or downwards, but they do not directly affect the hidden state itself. For example, a seasonality dummy could increase the purchase probabilities for ice cream products in the summer and decrease these probabilities in the winter, without affecting the customer's latent preferences. Similarly, promotion variables could increase the output probabilities of products in promotion. In our application, we add a single network layer with a rectified linear unit (ReLU) activation function. The resulting prediction for the next basket is then given by

$$\hat{\mathbf{b}}_t = \sigma(\mathbf{W}_{ba} \cdot \mathbf{a}_t + \mathbf{W}_{bx} \cdot \max(\mathbf{0}, \mathbf{W}_{bx} \cdot \mathbf{x}_t + \mathbf{e}_x) + \mathbf{e}_b). \tag{7}$$

The final prediction is now a function of customer preferences and observed covariates $\mathbf{x}_t$. In general, the branch that introduces covariates into the network can be extended with more layers, different activation functions or several branches depending on the type of covariates included. If desired, it can also allow for interactions between the observed covariates and the hidden state.

3.4. Parameter estimation

All model parameters of the GRU-based model are trained by minimizing an average binary cross-entropy loss function, defined as

$$\mathcal{L}(\mathcal{W}) = \frac{-1}{\sum_{c=1}^{C} N(T_c - 1)} \times \sum_{c=1}^{C} \sum_{t=2}^{T_c} \sum_{n=1}^{N} \left[b_{tn}^c \log\left(\hat{b}_{tn}^c(\mathcal{W})\right) + \left(1 - b_{tn}^c\right) \log\left(1 - \hat{b}_{tn}^c(\mathcal{W})\right) \right]. \tag{8}$$

where $\mathcal{W}$ is the set of all weights and biases in the network. b_{tn}^c refers to the nth element of the vector that represents the tth basket of customer c, which is either 1 (if product n is purchased) or 0. $\hat{b}_{tn}^c$ refers to the model's estimated probability that product n will be purchased by customer c in the tth basket, based on all prior baskets of the customer. Thus, $\hat{b}_{tn}^c$ is a function of the model parameters $\mathcal{W}$. The first basket of each customer is left out of the loss function, because it is not predicted (it only serves as input for future predictions). In more statistical terms, minimizing this loss function is equivalent to maximizing the log likelihood of N Bernoulli distributed dependent variables for each basket of each customer.

The loss function is iteratively minimized by backpropagation through time using the Adam optimization algorithm (Kingma & Ba, 2014). Adam is a variation of gradient descent where the learning rate is separately adapted for each parameter based on estimations of the first and second moment of the gradient. By doing this, strong oscillations in the loss function over iterations are prevented. A parameter update step is based on the gradient calculated over a mini-batch of observations. The training procedure is stopped once loss values have stabilized.

3.5. Hyperparameter tuning and regularization

Before optimization, the dimension of the latent state, d, must be chosen. Usually, this value is optimized by trying a range of values and selecting the dimension that yields the lowest out-of-sample loss after optimization. A larger latent state makes the model more flexible, but also increases training time and the risk of overfitting. The optimal latent state dimension is likely to differ per application. Other architecture choices, such as the choice of activation functions, could be tuned in a similar way.

Overfitting happens when the optimization algorithm starts updating the model parameters according to irregularities in the training data, rather than general purchase patterns. In some applications, the training set will be large enough to prevent the network from overfitting without using any regularization. This is the case in our first application, on the Instacart dataset. In our second application, on the smaller Dunnhumby dataset, we noticed cases of overfitting, harming out-of-sample performance. To detect overfitting, it is recommended to track out-of-sample loss values during training, next to the training loss. Training can be stopped early if out-of-sample loss values no longer decrease (or even start increasing).

An alternative, and more effective way to combat overfitting is applying dropout regularization (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). In general, the goal of regularization is to avoid that certain parts of the network specialize to (almost perfectly) predict single, or a few, observations. Dropout regularization achieves this by randomly setting a fixed percentage of node outputs to zero in each training step, effectively removing all information contained in these nodes. A node is defined as a single computational unit; together, a set of nodes forms a dense network layer. By randomly turning off different nodes in each training step, every node is forced to produce a meaningful, robust information signal by itself, without depending too much on the presence of information signals produced by other nodes in the network. Dropout therefore prevents the network from learning a set of strongly interdependent

weights that pick up insignificant, noisy features in the training data. To be specific, we advise to apply dropout in the input gate of the GRU in cases of overfitting, as proposed by Semeniuta, Severyn, and Barth (2016). In their approach, the set of inactive nodes randomly differs across customers within the same batch, but is fixed over each customer's sequence of baskets, such that the GRU does not lose the ability to learn long-term patterns. While evaluating the network on a holdout set, dropout should be turned off and all weights should be corrected for the dropout rate used during training (Srivastava et al., 2014). We also apply this type of dropout in our application on the Dunnhumby dataset.

4. Empirical application

In this section, we introduce the two datasets on which we will apply our GRU-based model. We first describe some data preprocessing steps, after which we describe our hyperparameter settings, performance measures and benchmarks.

4.1. Data

The first dataset we use comes from the American grocery delivery service Instacart[1] (Instacart, 2017). It contains information about approximately 3.4 million orders made by 206,209 customers in 2017. Each of these customers made at least four orders. The company offers 49,688 different products, organized in 134 aisles, which are in turn organized in 21 departments. Each order (or basket) contains one or more of these products. The exact date and time of the order is unknown, but day of the week and time of the day are given. Also, the time since the last purchase is known. We will use this information as basket-specific covariates.

Our second dataset comes from Dunnhumby,[2] a customer data science company. It contains all baskets of 2500 households at a retailer over a two-year period. The assortment consists of 92,399 products. Compared to the Instacart dataset, more baskets are observed here per customer. The exact calendar timing of the purchases is given, which we can include as basket-specific covariate.

4.1.1. Preprocessing basket data

The distribution of product purchase frequencies in both datasets is highly skewed, with a relatively small set of products representing the majority of purchases. For example, the most popular product in the Instacart dataset, (*Bananas*), is bought nearly 500,000 times in total, whereas the vast majority of products is bought no more than 200 times. If certain products appear extremely infrequently in the training data, it will be difficult for any model to learn general purchase patterns for these products. Instead, estimation will likely overfit model parameters to noisy features in the small number of baskets in which these products appear.

Commonly used approaches to solve this are removing all products appearing less frequently than a pre-determined threshold from the data or grouping the infrequently purchased products with other products. Details of this product aggregation procedure are provided in Appendix B. For the Instacart data, we set the minimum number of purchases for a product to 500 (0.015% of the total number of baskets) and apply the grouping procedure to create new product groups that are purchased at least 500 times. Applying this grouping procedure reduces the total number of products from 49,688 to 9407, an 81% reduction. The SKUs with less than 500 purchases, that are now grouped with others, together only comprised 11% of all original purchases. The result is still a relatively high-dimensional product assortment, with purchases of both slow-moving and fast-moving goods.

For the Dunnhumby dataset, we remove products purchased less than 50 times. This reduces the number of products from 92,339 to 10,370. We set the threshold for the Dunnhumby dataset lower, because this dataset is smaller than Instacart's and a higher threshold would remove too many products. We adopt the naive approach here to demonstrate that the model also works well with this type of preprocessing. The removed products together made up 20% of all original purchases. Finally, we removed customers with less than 3 baskets (0.9%) and capped sequences longer than 100 baskets at 100 (36.8%). Training on longer sequences hardly improves performance, whereas memory requirements and computation time grow substantially.

The data is split into a training set, used to obtain model estimates; a validation set, used to tune the model hyperparameters such as the size of the latent state; and a test set, used to evaluate the performance of our model. For the Instacart dataset, we exclude the last basket of 10,000 randomly sampled customers for validation purposes. For 10,000 other randomly sampled customers, we exclude the last basket for testing. All remaining baskets are assigned to the training set. For the Dunnhumby dataset, the last basket of each customer is excluded. Half of these (1239) is used for validation, the other half for testing. Characteristics of the resulting datasets are summarized in Table 1.

4.1.2. Basket-specific covariates

Basket-specific covariates can provide valuable information to improve the predictive performance of the model. The Instacart data provides two types of covariates we can use as covariate: time-of-day indicators and sequence-length indicators. As time indicators, we use the hour of the day and day of the week in which the purchase was made. This allows our model to infer which products are more (or less) likely to be bought at certain hours or days and adapt probabilities accordingly. We also include three sequence-length indicators: (i) the number of days between the current and previous basket transaction, (ii) the cumulative number of days between the current and first registered basket transaction and (iii) the number of previously purchased baskets. We expect these variables to influence the overall level of the purchase probabilities; as people shop more often and become more loyal, they might fill their baskets with more products (this is already suggested by the last column of Table 1). However, the most active customers might distribute their purchases over more shopping trips and therefore purchase less products per basket. Therefore, we do not expect the number of previous baskets to have a monotonically increasing effect on purchase probability levels.

For the Dunnhumby dataset, we consider the same covariates, but also add dummies for the month in which the purchase is made (this information is not available in the Instacart data). In this way, the model can learn to increase the purchase likelihood of season-specific products in the corresponding months. Price or promotion data is not available in either of the datasets, but could be informative in other applications. The included covariates are measured as follows:

- *Time-of-day indicators*: For each transaction, the day of the week (0–6) and hour of the day (0–23) are given. The distribution of basket purchases over a day is not equally distributed (Fig. 2 demonstrates this for the Instacart dataset). To have sufficient purchases in each time interval, the hour sets $\{21, 22\}$, $\{23, 0, 1, 2, 3, 4, 5\}$ and $\{6, 7\}$ are aggregated for each day. This provides us with $7 \times 16 = 112$ dummies for each day-hour combination. For identification, the last dummy (Sunday 23.00) is left out and the remaining dummies enter the last layer after first passing through a 3-dimensional linear layer. This layer imposes a factor structure on the impact of the hour dummies.

[1] https://www.kaggle.com/c/instacart-market-basket-analysis

[2] https://www.kaggle.com/datasets/frtgnn/dunnhumby-the-complete-journey

Table 1
Characteristics of the Instacart and Dunnhumby datasets. A purchase is defined as a certain customer buying a certain product at a certain time.

Dataset	# cust.	# unique products	# baskets	# purchases	avg. # baskets per cust.	avg. basket size
Instacart						
Training set	206,209	9,407	3,139,874	31,538,507	15.23	10.04
Validation set	10,000	9,407	10,000	104,418	1.00	10.44
Test set	10,000	9,407	10,000	104,725	1.00	10.47
All	206,209	9,407	3,159,874	31,747,650	15.32	10.05
Dunnhumby						
Training set	2,478	10,370	161,459	1,385,642	65.16	8.58
Validation set	1,239	5,122	1,239	11,280	1.00	9.10
Test set	1,239	5,018	1,239	11,409	1.00	9.21
All	2,478	10,370	163,937	1,408,331	66.16	8.59

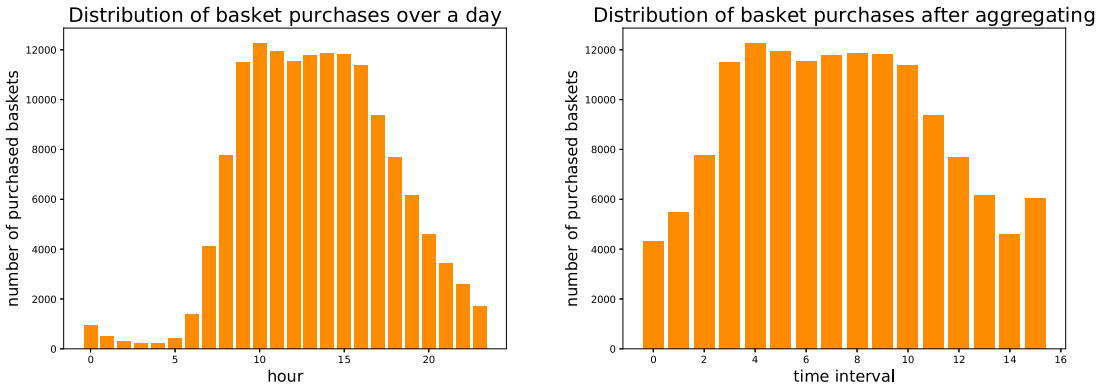


Fig. 2. Distribution of Instacart basket purchases (a) over 24 hours, (b) over intervals where evening, night and morning hours are merged.

- *Sequence-length indicators*: All three variables are divided by their respective maximum value to yield input values between 0 and 1. These variables enter the network in a different branch separate from the hour dummies, as we do not expect interactions between the two sets of variables. The sequence-length indicators first pass through a 5-dimensional ReLU layer, to allow for non-linearities, before entering the final layer.
- *Monthly dummies*: For the Dunnhumby dataset, the number of days since the start of the observation period is known for each basket purchase. These day numbers are consistent over customers, so baskets from different customers with the same day number correspond to the same calendar day in reality. The exact date these day numbers correspond to are not known, however. Although this means that we cannot assign purchases to the exact calendar month, we can divide the day numbers into consecutive 30/31-day periods, such that 12 of these periods make up exactly 365 days. We do so for each of the two years in the dataset. These manually created months are converted to 12 dummy variables, the last one of which is left out for identification. After initial experimentation, we concatenated the monthly dummies with the sequence-length indicators and, together, passed them through the 5-dimensional ReLU layer.

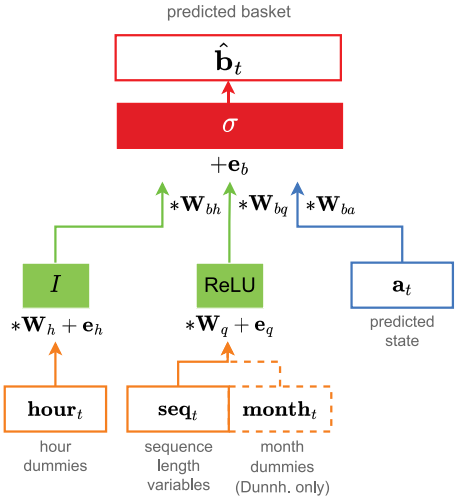


Fig. 3. Last layer of our model with covariates used for the Instacart application. *I* stands for a dense layer with linear (identity) activation, *ReLU* stands for a dense layer with ReLU activation.

Fig. 3 gives an overview of the last network layer in our empirical application including the branches corresponding to the hour dummies, sequence-length indicators and month dummies.

4.2. Hyperparameter settings

During training, each mini-batch consists of the complete sequences of 64 customers. We do not apply dropout in the Instacart application, as we observed that the validation loss does not diverge from the training loss during training. In contrast, for the Dunnhumby application, we observed clear overfitting without any regularization, for all considered hidden state dimensions d. This is probably due to the fact that the Dunnhumby training set contains 20 times less baskets than the Instacart training set. Based on predictive performance on the validation set, we apply a 50% dropout rate in the input layer. This strongly mitigates overfitting, leading to higher out-of-sample performance. As slight overfitting is still observed in late training stages, we stopped training once validation accuracy started decreasing.

For the hidden state dimension of the network, we considered values of d = 100, 250, 500, 750, 1000, 1500 and 2000. For the Instacart

application, the network with a 750-dimensional latent state attained the lowest loss in the validation set, so we used $d^* = 750$ in the final model. For larger latent states, out-of-sample evaluated loss no longer improves while training time increases strongly. Following similar reasoning, we found $d^* = 1500$ to be optimal for the Dunnhumby application. To speed up training time of the model with covariates, we use the trained parameters of the base model to initialize the overlapping parameters in the extended model. All computations are performed using the *PyTorch* package in *Python*. The GRU network is trained on the *Lisa Compute Cluster*, supported by *SURFsara*. For the Instacart dataset, training the model on a single GTX-1080Ti GPU until convergence takes approximately 27 h. On similar hardware, training the model for the Dunnhumby dataset takes approximately 9 hours[3].

4.3. Performance measures

We evaluate the performance of our model on the test set using several measures. Our target is to predict the last observed basket for all customers in the test set. The model outputs a purchase probability for each product in the assortment. In order to calculate the accuracy of these predictions, we first select the k^c highest ranked products from each customer's basket prediction. We set $k^c = N^c_{T_c}$, the number of products in the corresponding test basket of customer c. Afterwards, we count how many products in each test basket are contained in the predicted selection, and divide this number by k^c. This accuracy is averaged over all test baskets. In practice, the top-ranked products are usually the products that end up being recommended to a customer, so this measure indicates how well our model would perform in a practical recommendation setting. We also evaluate the precision and recall of our predictions in case our recommended set is smaller ($k^c = \lceil N^c_{T_c}/2 \rceil$) or larger ($k^c = 2N^c_{T_c}$) than the actual test basket. Precision refers to the proportion of the recommended set that is contained in the test basket, whereas recall refers to the proportion of the test basket that is contained in the recommended set. In case $k^c = N^c_{T_c}$, precision and recall are the same.

Besides looking at the top of our ranking, we also calculate the average predicted rank of the products purchased in the test basket. The product with highest predicted probability receives rank 1, such that a lower average rank indicates that the model can accurately distinguish products that are relevant for a customer from those that are not. We first calculate the average rank per test basket, before averaging over all test baskets, in order to guarantee equal contribution of large and small baskets in this measure.

4.4. Benchmarks

We compare the performance of the GRU-based model to five benchmarks. Our first three benchmarks are naive benchmarks, based on relatively simple heuristics. For the first benchmark, we rank products based on their general purchase frequency in the training data. Secondly, we rank products based on personal frequency, i.e. the number of appearances in the history of the same customer. Finally, we simply take the last training basket of each customer and use it as a prediction for the test basket. In case of ties, we rank the tied products based on their general frequency.

For the fourth benchmark, we use the state-of-the-art TARS next-basket prediction model (Guidotti et al., 2017), as described in Section 2. For the implementation of this method, we use the publicly available code by the authors.[4] We set the minimum number of prior baskets per customer to 1, in order to obtain predictions for all test customers. TARS identifies frequently occurring sequences in a customer's basket sequence and counts active sequences in order to assign scores to the products in the assortments. We rank the predictions based on these scores. Again, in case of ties, products are ranked based on their general frequency.

The final benchmark consists of a linearized version of the GRU network. We do so in order to evaluate the importance of having flexible, non-linear update functions for the latent state. In the linearized version, the forget gate is replaced by a fixed matrix multiplication. The input gate remains additive, but the newly observed basket and prior latent state are only multiplied by a fixed matrix before being added to the latent state. As a result, all non-linear updates are lost, and the update function becomes linear.

5. Results

In this section, we present the results of our next-basket prediction model with GRU dynamics, applied to the Instacart and Dunnhumby data. First, the predictive performance of our approach is compared to the benchmarks on a diverse range of performance measures. Afterwards, we discuss the scalability of the GRU compared to TARS. Finally, we provide some insights into the patterns identified by the GRU-based dynamic model.

5.1. Predictive performance

For the Dunnhumby dataset, we noticed that TARS was very slow in making predictions. As we will further demonstrate in Section 5.2, prediction time for TARS grows exponentially when there is high variability in a customer's purchase sequence. As the Dunnhumby dataset contains relatively long sequences with many unique purchases, making predictions for the 1239 test customers is estimated to take over 1000 hours of computation time on a single CPU. As we considered this infeasible in practice, even with parallel computing, we only used a maximum of 50 most recent baskets in order to make the TARS predictions for the Dunnhumby application. This reduced computation time to approximately 100 hours in total (compared to 9 training hours for the GRU).[5]

Tables 2 and 3 report the predictive performance of our approach and the five benchmarks on the Instacart and Dunnhumby test set, respectively. In general, predictive accuracy is lower for all models on the Dunnhumby dataset, most likely due to stronger variability in the purchase behavior of these households. Standard errors for the Dunnhumby dataset are also substantially higher, because of the smaller test set. Other than that, relative performance differences are very similar on both datasets.

We report the performance of the GRU-dynamics-based basket predictions with and without basket-specific covariates. Both reach very similar performance on all measures, but the model with covariates performs slightly better. This performance increase is minor and statistically insignificant for each of the separate performance measures, on both datasets. Apparently, most relevant relations between time of purchase and the set of purchased products are already captured through prior purchase behavior. However, the slight improvement in every single measure suggests that the inclusion of covariates would be beneficial when applied on a large scale and/or with more informative (marketing-mix) variables.

When comparing the GRU-based model to the benchmarks, we observe that all benchmarks are outperformed on nearly all measures. In both applications, only the personal frequency benchmark performs

[3] We published our code on https://github.com/luukvanmaasakkers/nextbasketpredictionGRU

[4] https://github.com/GiulioRossetti/tbp-next-basket

[5] To ensure our findings are not driven by the limited access to data for TARS, we also trained our GRU-based model on the 50 most recent baskets of each sequence. This only resulted in small performance drops compared to the reported results of the GRU-based model trained on the full sequences, and did not affect any of the conclusions drawn here.

Table 2
Predictive performance of the GRU and benchmarks on the Instacart dataset. Reported measures are the precision and recall for predicted sets with varying size, and the average rank of purchased products in the predictions. For $k^c = N^c_{T_c}$, precision and recall are identical. Standard errors are between brackets.

	k^c	GRU	GRU + cov.	General freq.	pers. freq.	Last basket	TARS	linear. GRU
Precision (%)	$\lceil N^c_{T_c}/2 \rceil$	38.52	**38.55**	8.91	36.62	30.03	35.20	9.91
		(0.32)	(0.31)	(0.14)	(0.31)	(0.29)	(0.30)	(0.14)
Recall (%)	$\lceil N^c_{T_c}/2 \rceil$	21.13	**21.15**	4.72	20.05	16.52	19.28	5.20
		(0.19)	(0.19)	(0.08)	(0.18)	(0.18)	(0.18)	(0.08)
Precision (%)	$N^c_{T_c}$	30.48	**30.54**	6.88	29.42	24.56	28.20	8.28
		(0.24)	(0.24)	(0.10)	(0.23)	(0.23)	(0.23)	(0.11)
Precision (%)	$2N^c_{T_c}$	20.11	20.21	4.91	**20.68**	15.61	18.42	6.15
		(0.14)	(0.14)	(0.06)	(0.14)	(0.13)	(0.14)	(0.07)
Recall (%)	$2N^c_{T_c}$	40.23	40.42	9.82	**41.36**	31.22	36.85	12.30
		(0.27)	(0.27)	(0.12)	(0.27)	(0.26)	(0.27)	(0.14)
Average rank		486.5	**480.6**	1971.6	960.3	1501.6	1370.2	1509.9
		(5.6)	(5.5)	(13.2)	(10.7)	(12.2)	(12.0)	(11.4)

Table 3
Predictive performance of the GRU and benchmarks on the Dunnhumby dataset. Reported measures are the precision and recall for predicted sets with varying size, and the average rank of purchased products in the predictions. For $k^c = N^c_{T_c}$, precision and recall are identical. Standard errors are between brackets.

	k^c	GRU	GRU + cov.	General freq.	pers. freq.	Last basket	TARS	linear. GRU
Precision (%)	$\lceil N^c_{T_c}/2 \rceil$	16.11	**16.69**	4.70	15.98	9.96	15.69	5.21
		(0.74)	(0.76)	(0.32)	(0.73)	(0.59)	(0.75)	(0.38)
Recall (%)	$\lceil N^c_{T_c}/2 \rceil$	10.21	**10.59**	2.53	9.79	6.21	9.70	3.24
		(0.58)	(0.59)	(0.19)	(0.66)	(0.45)	(0.55)	(0.32)
Precision (%)	$N^c_{T_c}$	13.69	**13.72**	3.93	13.16	8.07	12.33	4.70
		(0.63)	(0.64)	(0.23)	(0.60)	(0.49)	(0.61)	(0.34)
Precision (%)	$2N^c_{T_c}$	9.49	9.54	6.20	**9.74**	5.52	8.17	4.11
		(0.37)	(0.37)	(0.35)	(0.37)	(0.26)	(0.34)	(0.24)
Recall (%)	$2N^c_{T_c}$	18.98	19.09	12.40	**19.48**	11.04	16.30	8.24
		(0.73)	(0.74)	(55.64)	(0.74)	(0.54)	(0.69)	(0.48)
Average rank		1456.0	**1449.0**	2513.5	1688.2	2369.6	2261.5	2362.0
		(43.8)	(43.9)	(55.6)	(51.1)	(55.6)	(55.2)	(53.0)

better and only when the recommended set is twice as large as the test basket. This would indicate that the top of the ranking is more accurate for the GRU, but the personal favorites are more accurate in the sub-top. For the remaining performance metrics, the GRU outperforms all other benchmarks. The differences with the benchmarks are all significant on a 5% significance level for the Instacart dataset based on pairwise t-tests on the sample of basket-specific precision, recall and rank differences. For the Dunnhumby dataset, the personal frequency and TARS benchmarks are not significantly different from the GRU in terms of precision and recall, but only in average rank. A probable reason for the statistical insignificance of these differences is the lower number of observations in the Dunnhumby test set. For the remaining benchmarks, all performance differences with the GRU are significant.

The low average rank of the GRU indicates that the GRU is much more accurate in the tail of the ranking. This pattern is also confirmed by the ROC curves in Fig. 4. Here, all customer–product pairs in the Instacart test set are ranked from left to right, from highest to lowest predicted purchase probability, for each predictor. On the vertical axis, the cumulative percentage of actual purchases is counted as one goes down the ranking. Fig. 4(a) shows the entire curve, whereas 4(b) zooms in on the top 0.5% of pairs. These are the customer–product combinations for which the models are most confident that they will occur. The GRU with covariates is not separately displayed, as its ROC curve follows approximately the same path as the standard GRU-based model. Similar curves can be obtained for the Dunnhumby dataset, but are left out for conciseness. The curves of TARS and the last basket benchmark have clear kinks in their curves. These can be explained by the fact that these predictors only output positive scores for a relatively small subset of the products. For the remaining products, the ranking is based on the overall frequency, which is a much worse predictor. The GRU performs best in the top of the ranking, followed by the personal frequency benchmark, TARS and the last-basket benchmark. At some point, the GRU is overtaken by the personal frequencies, but it dominates all benchmarks again in the tail of the distribution. The linearized GRU predicts very poorly overall: it can hardly outperform the general frequency benchmark and is outperformed by the other two naive benchmarks. This shows that the non-linearity of the GRU is a necessary model component in order to reach high accuracy.

The used datasets contain relatively many repeat purchases, resulting in a strong performance of the personal frequency benchmark. In the Instacart data, 60% of products purchased in test baskets are already purchased before by the same customer. For the Dunnhumby dataset, this percentage is 51%. In contexts with more variation in purchased baskets, this benchmark is likely to perform much worse compared to more sophisticated models.

For further analysis, we report results only for the Instacart dataset. Comparable analyses on the Dunnhumby dataset can be obtained from the authors. To investigate performance differences further, we split the test baskets in 10 equally large groups with an increasing number of prior baskets. For these 10 groups, we evaluate the precision of the GRU, TARS and personal frequency benchmark for $k = N^c_{T_c}$. The results are presented in Fig. 5. If the TARS or personal frequency benchmark performs significantly different from the GRU in a decile, this is indicated by asterisks. In general, all predictors benefit from longer prior sequences, as this provides more information about the customer's preferences. For short sequences, the GRU has trouble outperforming the personal frequency benchmark. For larger sequences, however, the performance gap rapidly increases. As expected, TARS performs relatively poor for short purchase histories, but can also outperform the benchmark for longer sequences. Nevertheless, the GRU significantly dominates TARS for all groups in terms of precision. This indicates that recommending personalized favorites is a good alternative for new customers, but for customers with longer histories, using sophisticated models results in a considerable increase in performance. Moreover, by

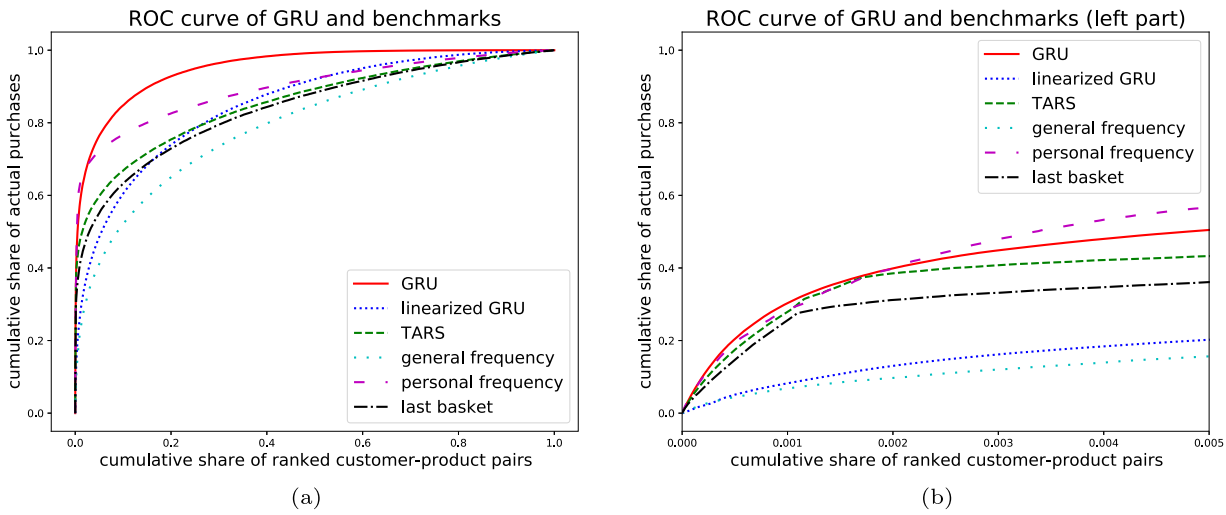


Fig. 4. ROC curve of the GRU-based model compared to benchmarks. All predicted customer–product pairs in the test set are ranked in descending order from left to right for each model, and the cumulative share of actual purchase pairs is reported on the vertical axis. Panel (a) shows the entire curve, panel (b) is zoomed in on the top 0.5% pairs.

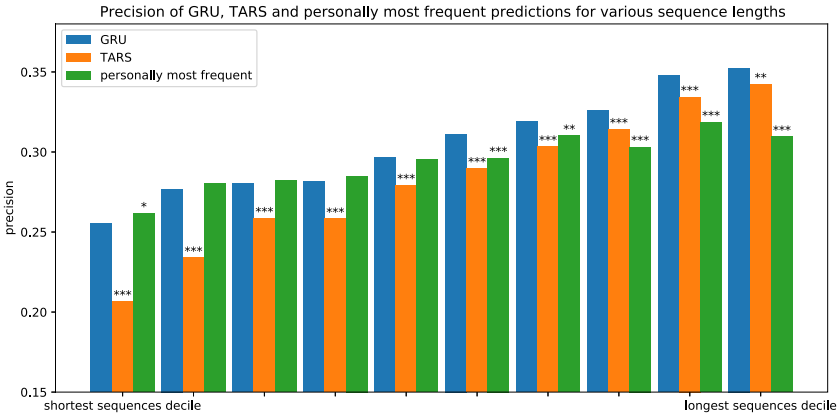


Fig. 5. Precision of GRU, TARS and the personal frequency benchmark for $k = N_T^c$, for subsets of test baskets with increasing number of prior baskets. The first decile contains only test sequences of length 4 (the last of which is the test basket). The last decile contains test sequence of length 36 and higher. Asterisks indicate whether the performance difference with respect to the GRU is significant using a t-test on the sample of basket-specific precision differences (at * = 10%, ** = 5% or *** = 1% significance level).

only recommending personal favorites, a customer will never see new items, trapping customers in a so-called 'similarity hole'. This is generally not desirable, as recommenders often also have an informative purpose (Herlocker, Konstan, Terveen, & Riedl, 2004).

5.2. Scalability

Comparing the training time of the GRU with TARS is not straightforward, as both methods are very different in nature. The GRU requires a relatively long training stage (27 hours for the Instacart dataset, 9 hours for the Dunnhumby dataset in our application), in which its parameters are optimized. Afterwards, the model can make predictions for thousands of customers within a few seconds. TARS, on the contrary, treats each customer sequence separately. It first identifies frequently occurring patterns in the customer's purchase history and then counts the active recurring sequences in order to make predictions. If a customer has a richer purchase history with more unique product purchases, the number of potential recurrent sequences becomes larger and therefore the time required to make a prediction increases. Fig. 6 shows the average computation time per test basket of TARS in seconds for 10 subsets of test baskets, with an increasing number of unique product purchases in their history. For short histories with few unique products, TARS can make predictions within milliseconds. However, for larger histories with many unique product purchases, a prediction can take minutes. The top decile contains histories with over 130 unique product purchases, which is not uncommon in practice. For the most extreme case in our application, the prediction took 28 minutes for a single basket prediction. In total, training the models and making predictions for our 10,000 test customers took 2 hours longer for TARS than for the GRU, on similar hardware. If the number of test customers would double, the GRU would require a few more seconds for its predictions, because it is already trained. On the other hand, the computation time of TARS would also double. For that reason, the GRU is better scalable to longer sequences and larger assortments than TARS, making it more suitable to apply in practice.

5.3. Model interpretation

After training the network, we can further evaluate its properties by analyzing the trained weights, biases and hidden states. This will help understand which properties of purchase behavior are captured by the different components of the model. In this section, we perform such analyses on the Instacart dataset. Comparable analyses on the Dunnhumby dataset can be obtained from the authors. We interpret the hidden state as a lower-dimensional summary of a customer at a certain point in time, based on their complete past purchase behavior. To illustrate how hidden states of customers evolve over time, we map the 750-dimensional latent states of 100 randomly selected test

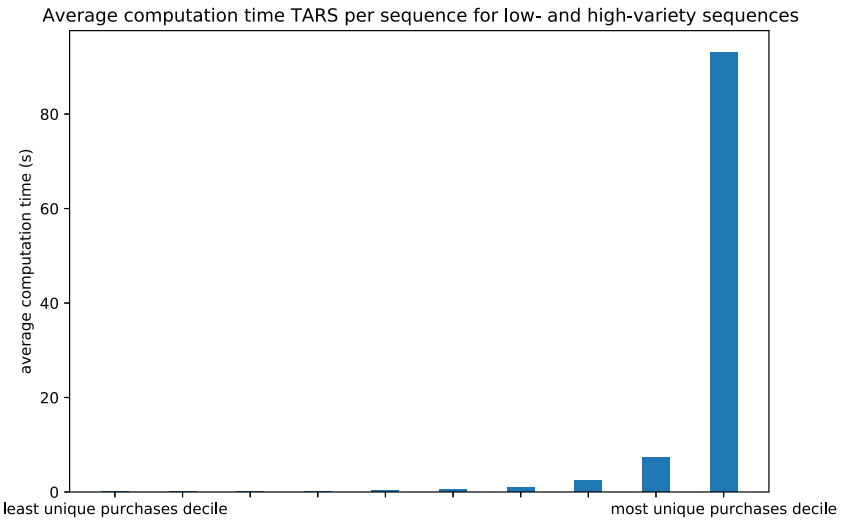


Fig. 6. Computation time of TARS in seconds for subsets of test baskets with increasing number of uniquely purchased products in their history.

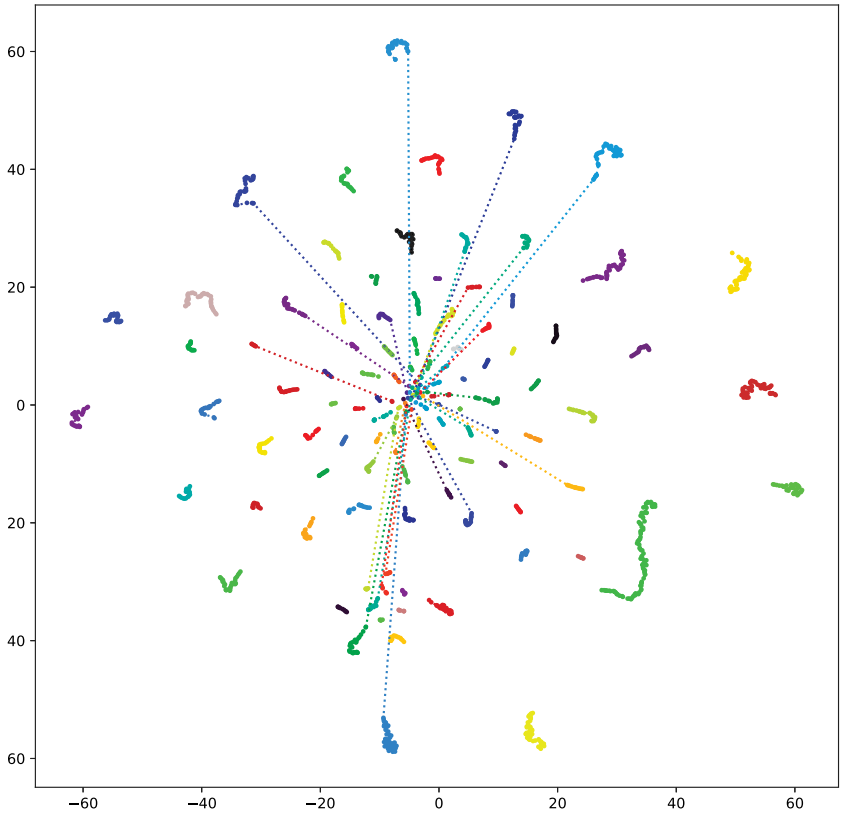


Fig. 7. TSNE mapping of hidden states of 100 randomly selected test users. Each point represents a two-dimensional representation of a single hidden state. Hidden states of the same customer have the same color and are connected by a dotted line.

customers into two dimensions using t-stochastic neighbor embedding (TSNE, Van der Maaten and Hinton (2008)). This mapping does not use the information on which latent states correspond to which customer. After obtaining the mapping, we use different colors for the latent states of each customer and connect consecutive hidden states by a dotted line. This allows us to investigate (dis)similarities in latent states between customers and over time (see Fig. 7).

It turns out that hidden states are nearly perfectly clustered by customer (even though the TSNE algorithm does not use customer ID information). The dotted line is often not visible, as consecutive hidden states are already very close together in the map. This indicates clear heterogeneity in the learned preferences within the customer base. Variation of hidden states within customers is much smaller, but also clearly present. For many customers, we observe 'jumps' between the early hidden states from the center of the map towards the outer ranges. This is explained by the model initially needing to learn the customer's preferences. In the center of the map, hidden states are close to the mean. In the outer ranges, hidden states deviate far from the mean and therefore exhibit more specific preference. With limited consumption histories available, the estimated preference of a customer by the model can quickly change when new information comes in, explaining these large jumps from the center to the outside. However, also for longer

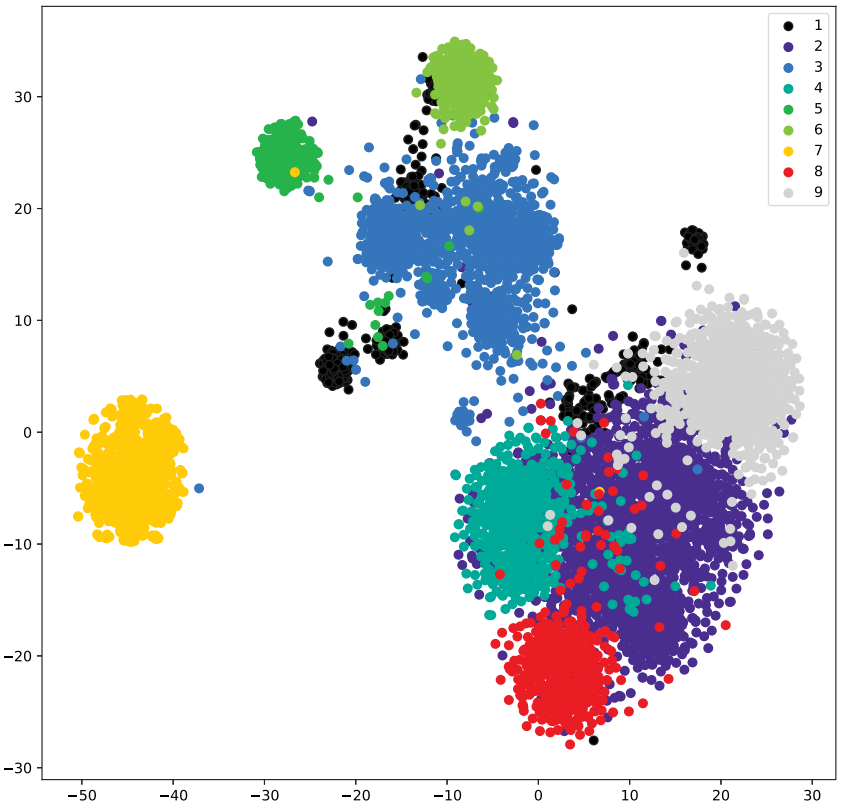


Fig. 8. TSNE mapping of the final hidden states of all 10,000 test customers. Colors are assigned to 9 clusters formed by hierarchical clustering on the full dimension of the hidden state.

Table 4
Top 5 aisles that are purchased most frequently in each cluster from Fig. 8, relative to average aisle purchase frequencies over all customers.

1	2	3
Water seltzer sparkling water	Frozen vegan vegetarian	Marinades meat preparation
Energy granola bars	Tofu meat alternatives	Refrigerated pudding desserts
Tea	Vitamins supplements	Hot dogs bacon sausage
Candy chocolate	Prepared meals	Cocoa drink mixes
Cream	Frozen breads doughs	Frozen juice
4	**5**	**6**
Packaged seafood	Spirits	Beauty
Meat counter	Specialty wines champagnes	First aid
Grains rice dried goods	Beers coolers	Eye ear care
Oils vinegars	Red wines	Body lotions soap
Fresh herbs	White wines	Baby accessories
7	**8**	**9**
Packaged produce	Poultry counter	Specialty cheeses
Trail mix snack mix	Seafood counter	Bread
Frozen juice	Packaged poultry	Eggs
Bulk dried fruits vegetables	Indian foods	Milk
Mint gum	Skin care	Bulk grains rice dried goods

sequences, we observe hidden states of customers moving slowly but substantially through the map, indicating changes in preferences over time.

Upon closer inspection, we observe that some individual hidden state *elements* are nearly constant over time, whereas others show more variation. This indicates that parts of the hidden state pick up habit persistence, whereas others pick up variety seeking behavior. The fact that some hidden state paths cover larger distances than others could be an indication that some customers exhibit more variety seeking behavior than others.

To further examine heterogeneity in the customer base, we cluster the final hidden states of all 10,000 test customers with hierarchical clustering (Johnson, 1967), based on Euclidean distances between the hidden states. We found that using 9 clusters yields intuitive groups of customers. To visualize the clusters in two dimensions, we mapped these hidden states again with TSNE, without using the clusters, in Fig. 8 and afterwards color them based on the assigned clusters. To illustrate how the clusters differ in purchase behavior, we show the relatively most frequently purchased aisles in the test baskets of the customers in each cluster in Table 4. These are the aisles for which purchase frequencies within the cluster differ the most from the average purchase frequencies over all customers. Note that these test baskets were not used to calculate the hidden states.

Some clear, intuitive groups of customers become visible. Vegan products are purchased often by customers in cluster 2, whereas clusters 4 and 8 consist of customers that purchase many meat products.

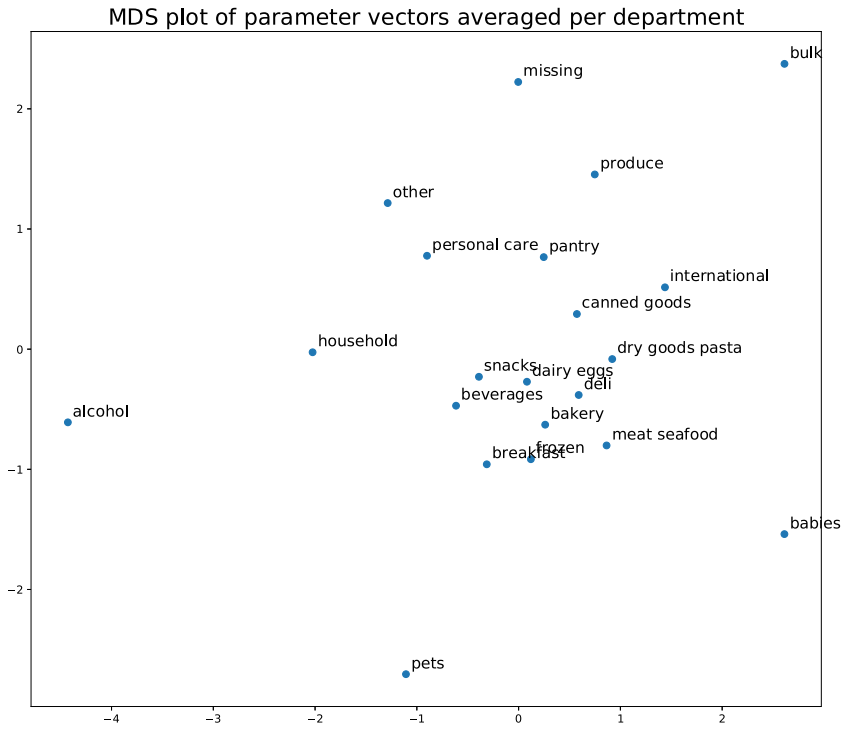


Fig. 9. Low-dimensional representation of average weight vectors for each product category, plotted using multi-dimensional scaling. Each point corresponds to a category, with corresponding label. Nearby categories have more similar embeddings than categories far apart.

Cluster 5 contains people consuming many alcoholic drinks, while cluster 6 purchases many beauty and baby products. Fruits and vegetables are popular in cluster 7, whereas breakfast products are often purchased in cluster 9. The fact that the reported aisles are purchased in the test basket, which has not affected the final hidden states upon which customers are clustered, also gives clear indication that the hidden states contain strong predictive information about future purchases.

If we interpret the hidden states as representations of customer preferences, we can regard the $d \times N$ weight matrix between the hidden state and the last sigmoid layer as a description, or *embedding*, of the N products, such that the multiplication of a customer's hidden state and this weight matrix provides the utility this customer obtains in their current state from each product. Following this reasoning, see also Gabel and Timoshenko (2021), similar products should have similar d-dimensional embeddings. We check this by considering the category labels that were assigned by Instacart to all products, but that were not used during the training process.

In Fig. 9, the product embeddings are averaged within each category and plotted in a two-dimensional space using multi-dimensional scaling. Categories close to each other have similar weight vectors and thus similar predicted purchase behavior. Some categories appear to be far away from other categories, such as *alcohol*, *pets*, *babies* and *bulk*. This means that purchase behavior regarding these categories is very different from each other and also from the other categories. Especially categories that do not comprise regular daily food are found to be different, whereas most food categories are clustered together in the center. The *bulk* category contains food products that are packed in very large quantities, so it makes sense that purchase behavior regarding this category is also found to deviate from other food categories. Overall, these patterns illustrate that the obtained weights indeed identify a latent structure in the product assortment. In practical applications, studying this structure may provide valuable knowledge.

To get more insights into the learned patterns on the product level, we consider a customer who has purchased a single basket with a single product. Given this basket, we make a prediction of the next basket and see which products undergo the strongest relative increase in probability, compared to a prediction for a customer with no history. Four examples are given in Table 5. Note that in our data pre-processing, some products are clustered together into small groups. The product names reported here are the names of the most frequently purchased product of their group. In most cases, other products in the same group have a very similar name. The examples show that the repeat purchase probability is always predicted to be very high: the purchased product undergoes by far the strongest increase in probability for the next basket. This indicates strong habit persistent purchase behavior of customers. The remaining products that increase strongly in probability are usually similar products, or products that complement the previously purchased product. For example, several types of baby wipes receive high probability after a baby wipe purchase, but also several types of baby food. Similarly, a taco sauce purchase increases the probability of purchasing tortilla chips, coleslaw and diced tomatoes, which would complement each other in a dish.

6. Conclusion and discussion

In this study, we proposed a GRU-based network for next basket prediction for large product assortments. We showed that the GRU-based model outperforms several benchmarks in terms of predictive accuracy, including the state-of-the-art TARS next basket prediction model (Guidotti et al., 2017). We also demonstrate that its strong performance is caused by its flexible, non-linear operations. In both of our applications, a naive benchmark that predicts the most frequently purchased products in the customer's history performs remarkably well. This indicates habit persistence in the customers' purchase behavior, which is also reflected in the slowly changing preference states in our model. For short sequences, the naive benchmark performs similarly to the GRU-based model and is therefore a good, quick alternative for marketeers. When longer purchase histories are available, the GRU-based model dominates in terms of performance.

A drawback of the GRU-based model is that it requires hours of training time and a (preferably) large training set of observed basket

Table 5
Examples of how basket purchases containing one product affect the predicted probabilities of the next basket by the model. The reported percentage increase in probability is in comparison with a basket prediction without prior basket information.

First basket	Strongest probability increases for next basket	
Original Medium Taco Sauce	Original Medium Taco Sauce	+5492%
	Bite Size Tortilla Chips	+189%
	3 Color Deli Coleslaw	+188%
	Mild Diced Tomatoes & Green Chilies	+186%
	Basmati Rice	+186%
Vegan Burger Patties	Vegan Burger Patties	+15 340%
	Frankfurters Vegetarian	+574%
	Deli Slices, Bologna Style	+392%
	Lightly Seasoned Chick'n Scallopini	+391%
	Vegan Apple Maple Breakfast Sausage	+339%
Gluten Free Lasagna	Gluten Free Lasagna	+15 127%
	Gluten Free Spaghetti Pasta	+206%
	Gluten Free Cheddar Macaroni	+203%
	Rice Mac & Cheese	+200%
	Brown Rice & Vegetables Bowl	+180%
Free & Clear Baby Wipes	Free & Clear Baby Wipes	+6186%
	Free & Clear Size 4 Baby Diapers	+205%
	Free & Clear Unscented Baby Wipes	+202%
	Strawberry and Banana Fruit Puree	+137%
	Organic Apples, Carrots and Parsnips Puree	+134%

purchases. For smaller datasets, overfitting can become a problem, and model regularization is required. TARS is a viable alternative in case no training set is available, although TARS can also be time consuming when producing predictions for customers with a high variety in their purchases.

Including additional covariates led to a minor, statistically insignificant improvement in predictive performance. A possible reason for this is that the available covariates are not strong predictors of product purchase, given the already available purchase history. The added value of including marketing-mix variables such as prices and promotions in the network is an interesting direction for future research. For these types of variables, it might even be useful to provide them as input to the gated recurrent unit as well. In that way, memories of prior prices and promotions can be captured in the latent state, and through that way, current prices and promotions can have an effect on future purchase likelihoods as well.

In our basket representations, we did not capture the quantities of products purchased in a basket. One reason for this is that the managerial relevance of predicting purchase quantities is limited. For personalizing recommendations and promotions, it is most interesting to know which products a customer is interested in, rather than how much of a product a customer will purchase. Secondly, defining quantity is not straightforward, given that products can have different sizes or volumes. In a context where purchase quantity is expected to be important, for example when consumers stock pile products, future research can study the role of including quantity information to account for such inventory processes that could further improve predictive accuracy.

CRediT authorship contribution statement

Luuk van Maasakkers: Conceptualization, Methodology, Software, Data curation, Formal analysis, Investigation, Writing – original draft, Visualization. **Dennis Fok:** Conceptualization, Methodology, Investigation, Writing – review & editing, Supervision. **Bas Donkers:** Conceptualization, Methodology, Investigation, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data is publicly available, links are provided in the manuscript.

Appendix A. Summary table of notation

See Table A.6.

Appendix B. Pre-clustering of products

In this section, we describe the pre-clustering of products in the Instacart dataset. We restrict the clusters to contain only products that belong to the same aisle (as defined by Instacart). In this way, we ensure that clusters consist of similar products. The detection of co-occurrence patterns across aisles is left for the GRU-based model. Within an aisle, two products are assumed to be similar when they are bought together with similar sets of products from that same aisle. For example, if *Braeburn Apples* and *Fireside Apples* would both appear often in baskets together with *Bananas*, *Kiwis* and *Strawberries* (which are all from the *Fresh fruits* aisle), the two types of apples are considered similar. Note that *Braeburn Apples* and *Fireside Apples* do not necessarily appear often together themselves to be considered similar.

Formally, for each aisle a, an $N_a \times N_a$ co-occurrence matrix $\mathbf{C}_a$ can be constructed, where N_a denotes the number of items in aisle a. Let $\mathcal{B}_i$ denote the set of baskets in which product i appears. The off-diagonal elements of a matrix $\mathbf{C}$ are then defined by

$$c_{ij} = \frac{|\mathcal{B}_i \cap \mathcal{B}_j|}{|\mathcal{B}_i|} \quad \text{for } i, j = 1, \dots, N_a, \quad i \neq j. \tag{B.1}$$

The values of the diagonal elements c_{ii} are debatable. To make sure that its value is not zero, but also not too high (and influential) compared to other values in the same row, it is set to $\frac{1}{|\mathcal{B}_i|}$. Next, an $N_a \times N_a$ cosine similarity matrix $\mathbf{S}$ is constructed over the rows of $\mathbf{C}$, where the elements of $\mathbf{S}$ are given by

$$s_{ij} = \frac{\mathbf{c}_i \cdot \mathbf{c}_j}{\|\mathbf{c}_i\| \|\mathbf{c}_j\|}, \tag{B.2}$$

where $\mathbf{c}_i$ is the ith row of $\mathbf{C}$. Finally, for each aisle separately, products are clustered in groups as follows:

1. Select the set of products corresponding to (clusters of) products with a purchase frequency lower than 500.

Table A.6
Explanation of notation used in the methodology. The superscript c for model in- and outputs is often left out in the methodology for convenience, but added here for completeness.

Indices	
c	Customer index, ranging from 1 to C
t	Relative time index of basket in a customer's sequence, ranging from 1 to T_c
n	Product index, ranging from 1 to N
Constants	
C	Number of unique customers
T_c	Number of baskets in customer c's purchase sequence
N	Number of unique products in the assortment
Model parameters	
d	Dimension of the GRU latent state (hyperparameter to be set before training)
$\mathbf{W}_s$	Multiplicative weight matrix in the scaling layer
$\mathbf{W}_f$	Multiplicative weight matrix in the forget layer
$\mathbf{W}_i$	Multiplicative weight matrix in the input layer
$\mathbf{W}_{bx}$	Multiplicative weight matrix in the output layer, multiplied with the output of the covariate branch
$\mathbf{W}_{ba}$	Multiplicative weight matrix in the output layer, multiplied with the hidden state
$\mathbf{e}_s$	Additive weight vector in the scaling layer
$\mathbf{e}_f$	Additive weight vector in the forget layer
$\mathbf{e}_i$	Additive weight vector in the input layer
$\mathbf{e}_b$	Additive weight vector in the output layer
$\mathcal{W}$	Set of all trainable model parameters (i.e. all parameters above except d)
Model input	
$\mathbf{b}_t^c$	N-dimensional binary vector representing customer c's basket at time t f
b_{tn}^c	nth element of $\mathbf{b}_t^c$, 1 if product n is purchased and 0 if not
$\mathbf{x}_t^c$	Vector representing customer c's basket-specific covariates at time t
(intermediate) model output	
$\mathbf{s}_t^c$	Output of the scaling layer for customer c at time t, a d-dimensional vector with continuous values between 0 and 1
$\mathbf{f}_t^c$	Output of the forget layer for customer c at time t, a d-dimensional vector with continuous values between 0 and 1
$\mathbf{i}_t^c$	Output of the input layer for customer c at time t, a d-dimensional vector with continuous values between −1 and 1
$\mathbf{a}_t^c$	Hidden state of customer c at time t
$\hat{\mathbf{b}}_t^c$	Predicted basket for customer c at time t, an N-dimensional vector with continuous values between 0 and 1
$\hat{b}_{tn}^c$	nth element of $\hat{\mathbf{b}}_t^c$, containing the estimated probability of customer c purchasing product n at time t
Activation functions	
σ	Sigmoid function
tanh	Hyperbolic tangent function
ReLU	Rectifier function

2. The product from this set of infrequently purchased products with the highest similarity with any other product (cluster) is grouped with that product. This could be a product that is already purchased more than 500 times.
3. For the newly clustered products, the corresponding rows and columns in **S** are replaced by the similarities that correspond to the newly created cluster of products. As the purchase frequency of the new cluster is the sum of the two single purchase frequencies, the purchase frequency of this product has increased.
4. The first three steps are repeated until no (cluster of) products with purchase frequency lower than 500 are left for the considered aisle.

This product aggregation procedure results in 9407 products or product clusters with an average size of 5. 23% of these product clusters contain exactly one of the original products, 56% contain three or less.

References

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on management of data*. pp. (207–216).

Bai, T., Nie, J.-Y., Zhao, W. X., Zhu, Y., Du, P., & Wen, J.-R. (2018). An attribute-aware neural attentive model for next basket recommendation. In *Proceedings of the 41st international ACM SIGIR conference on research & development in information retrieval* (pp. 1201–1204). ACM.

Bel, K., Fok, D., & Paap, R. (2018). Parameter estimation in multivariate logit models with many binary choices. *Econometric Reviews*, *37*(5), 534–550.

Bengio, Y., Simard, P., Frasconi, P., et al. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166.

Bronnenberg, B. J., Dubé, J.-P. H., & Gentzkow, M. (2012). The evolution of brand preferences: Evidence from consumer migration. *American Economic Review*, *102*(6), 2472–2508.

Chen, Y.-L., Tang, K., Shen, R.-J., & Hu, Y.-H. (2005). Market basket analysis in a multiple store environment. *Decision Support Systems*, *40*(2), 339–354.

Chintagunta, P. K. (1999). Variety seeking, purchase timing, and the "lightning bolt" brand choice model. *Management Science*, *45*(4), 486–498.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

Erdem, T. (1996). A dynamic analysis of market structure based on panel data. *Marketing Science*, *15*(4), 359–378.

Gabel, S., & Timoshenko, A. (2021). Product choice with large assortments: A scalable deep-learning model. *Management Science*.

Guidotti, R., Rossetti, G., Pappalardo, L., Giannotti, F., & Pedreschi, D. (2017). Market basket prediction using user-centric temporal annotated recurring sequences. In *2017 IEEE international conference on data mining (ICDM)* (pp. 895–900). IEEE.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 5–53.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Instacart (2017). The instacart online grocery shopping dataset. Accessed from https://www.instacart.com/datasets/grocery-shopping-2017 on March 26, 2018.

Jacobs, B. J., Donkers, B., & Fok, D. (2016). Model-based purchase predictions for large assortments. *Marketing Science*, *35*(3), 389–404.

Jacobs, B. J., Fok, D., & Donkers, B. (2021). Understanding large-scale dynamic purchase behavior. *Marketing Science*, *40*(5), 844–870.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, *32*(3), 241–254.

Kannan, P., & Sanchez, S. M. (1994). Competitive market structures: a subset selection analysis. *Management Science*, *40*(11), 1484–1499.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Koehn, D., Lessmann, S., & Schaal, M. (2020). Predicting online shopping behaviour from clickstream data using deep learning. *Expert Systems with Applications*, *150*, Article 113342.

Le, D.-T., Lauw, H. W., & Fang, Y. (2019). Correlation-sensitive next-basket recommendation. In *Proceedings of the 28th international joint conference on artificial intelligence*. pp. (2808–2814).

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(11).

McAlister, L. (1979). Choosing multiple items from a product class. *Journal of Consumer Research*, *6*(3), 213–224.

McAlister, L. (1982). A dynamic attribute satiation model of variety-seeking behavior. *Journal of Consumer Research*, *9*(2), 141–150.

McAlister, L., & Pessemier, E. (1982). Variety seeking behavior: An interdisciplinary review. *Journal of Consumer Research*, *9*(3), 311–322.

Montgomery, A. L., & Smith, M. D. (2009). Prospects for personalization on the internet. *Journal of Interactive Marketing*, *23*(2), 130–137.

Rendle, S., Freudenthaler, C., & Schmidt-Thieme, L. (2010). Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on world wide web* (pp. 811–820). ACM.

Russell, G. J., & Petersen, A. (2000). Analysis of cross category dependence in market basket selection. *Journal of Retailing*, *76*(3), 367–392.

Rust, R. T., & Chung, T. S. (2006). Marketing models of service and relationships. *Marketing Science*, *25*(6), 560–580.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on world wide web*. pp. (285–295).

Semeniuta, S., Severyn, A., & Barth, E. (2016). Recurrent dropout without memory loss. arXiv preprint arXiv:1603.05118.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.

Vilcassim, N. J., & Jain, D. C. (1991). Modeling purchase-timing and brand-switching behavior incorporating explanatory variables and unobserved heterogeneity. *Journal of Marketing Research*, *28*(1), 29–41.

Wang, P., Guo, J., Lan, Y., Xu, J., Wan, S., & Cheng, X. (2015). Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval* (pp. 403–412). ACM.

Yu, F., Liu, Q., Wu, S., Wang, L., & Tan, T. (2016). A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th international ACM SIGIR conference on research and development in information retrieval* (pp. 729–732). ACM.

Zhang, J., & Wedel, M. (2009). The effectiveness of customized promotions in online and offline stores. *Journal of Marketing Research*, *46*(2), 190–206.