Expert Systems With Applications 176 (2021) 114890

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A case study of batch and incremental recommender systems in supermarket data under concept drifts and cold start

Antônio David Viniski [a], Jean Paul Barddal [a,*], Alceu de Souza Britto Jr. [a], Fabrício Enembreck [a], Humberto Vinicius Aparecido de Campos [b]

[a] Graduate Program in Informatics (PPGIa), Pontifícia Universidade Católica do Paraná (PUCPR), Rua Imaculada Conceição, 1155 Curitiba, Brazil
[b] Himarket, Avenida Visconde de Guarapuava, 2764 Curitiba, Brazil

ARTICLE INFO

Keywords:
Collaborative filtering
Recommender systems
Positive-only feedback
Dataset

ABSTRACT

Recommender systems uncover relationships between users and items, thus allowing personalized recommendations. Nonetheless, users' preferences may change over time, the so-called concept drifts; or new users and items may appear, making the recommender system unable to accurately map the relationship between users and items due to the cold start problem. Consequently, concept drift and cold start are challenges that downgrade the recommender system's predictive performance. This paper assesses existing approaches for collaborative-filtering recommender systems over a real supermarket dataset that exhibits both of the issues mentioned above. For this purpose, our comparative analysis encompasses batch and streaming learning approaches. As a result, we can observe that streaming-based models achieve better recommendation rates since these are tailored to fit the concept drift. More specifically, the predictive performance of streaming-based recommendations increases by up to 21% over those provided by batch methods. The supermarket dataset used in experimentation is also made publicly available for future studies and recommender systems comparisons.

1. Introduction

Recommender systems are a hot topic in today's world. Businesses of all kinds are interested in implementing recommender systems, as these allow individualized interactions with customers based on their preferences. A recommender system predicts an item's probability to be preferred by a particular user (Zhang et al., 2019; Ricci et al., 2011). Over the years, different approaches were tailored to develop recommender systems. These are categorized as collaborative filtering (CF), content-based filtering (CBF), or hybrid approaches that combine the strategies mentioned above. Determining whether to use CF, CBF, or hybrid approaches depends on the availability and format of the data. In CF, the only data required is a list of user-item interactions, while in CBF, items' details are required. Consequently, CF is less restrictive and has been the target of many works over the years (Bobadilla et al., 2013; Zhang et al., 2019).

In this paper, we focus on two problems that affect recommender systems. The first is *concept drift*, which refers to changes in the data behavior over time (Tsymbal, 2004; Webb et al., 2018). In recommender systems, concept drift reflects changes in the interactions between customers and items, either because (i) customers' preferences change, (ii) new items become available for purchase, etc. The second is *cold start*, which occurs when new customers or items appear in the recommendation scenario. Such a problem is challenging, because the recommender model cannot make robust inferences for users or items about which it has not yet collected enough information (Ocepek et al., 2015; Shao et al., 2021).

Recommender systems are traditionally trained in a batch fashion, which means that given a training set composed of interactions between users and items, a static model is learned and deployed *ad eternum*. Consequently, it is relevant to tailor recommender systems that can be incremented over time, assuming that the interactions between users and items are made available as a stream of events.

In this paper, our goal is to bring forward a case study of existing recommender algorithms in a real-world supermarket scenario that exhibits both concept drifts and cold start problems. The contribution of this paper is threefold, as follows:

* Corresponding author.
E-mail addresses: adviniski@ppgia.pucpr.br (A.D. Viniski), jean.barddal@ppgia.pucpr.br (J.P. Barddal), alceu@ppgia.pucpr.br (A.S. Britto Jr.), fabricio@ppgia.pucpr.br (F. Enembreck), humberto.campos@himarket.club (H.V.A. Campos).

https://doi.org/10.1016/j.eswa.2021.114890
Received 1 May 2020; Received in revised form 2 March 2021; Accepted 6 March 2021
Available online 16 March 2021

- A comparison of existing filtering recommender systems in which the impact of concept drift and cold start is assessed in both batch and streaming fashions.
- Evidence that the incremental ability of the streaming-based recommender systems allows a better recovery when cold start is present.
- A novel real-word supermarket dataset that exhibits concept drifts and cold start problems is made publicly available.

This paper is divided as follows. Section 2 describes recommender systems, types of feedback, the problem of concept drift, and the challenge of cold start. Section 3 describes existing works on positive-only recommender systems in both batch and stream learning scenarios. Section 4 details a new dataset we make available regarding supermarket transactions, which exhibits concept drift and cold start characteristics. Section 5 describes the experiments undertaken to perform the proposed analysis of recommender systems. Section 6 analyzes existing works in recommender systems to answer whether streaming recommender systems overcome batch approaches w.r.t. concept drifts and cold start. Finally, Section 7 concludes this paper and describes future works.

2. Recommender systems

Recommender systems have been successfully applied in many real-world scenarios, proving efficient in handling diverse information related to users, items, and their interaction. The goal of recommender systems is personalization, which refers to the ability to recommend relevant items to specific users based on their past interactions with the system. Considering the type of information used in the recommendation process and how the system models the user-item relationship, recommender systems are categorized into three classes: Content-Based Filtering, Collaborative Filtering, and Hybrid Filtering.

Content-based filtering techniques recommend items to users considering items' characteristics and descriptions (Wei et al., 2017; Ricci et al., 2011). One central component of content-based approaches is the user modeling process that infers users' interest from items they interacted with. Items are compared with items previously liked by users, and the best-matched items are recommended (Beel et al., 2016).

The second approach, collaborative filtering, doesn't require previous knowledge of users or items, and thus, it is the most common approach for developing recommender systems. In practice, collaborative systems recommend items based on past user-item interactions (Nassar et al., 2020). The rationale behind collaborative filtering is that users who have expressed similar interests will share future interests. Thus, the items recommended for a particular user are related to items preferred by users that demonstrated similar interests previously (Portugal et al., 2018; Yin et al., 2019).

Hybrid filtering techniques combine content-based and collaborative filtering to make recommendations. Thus, the recommended items rely on both item descriptions and user-item interactions, thus enabling the recommendation system to find similarities between users, items, and user-item relationships (Portugal et al., 2018).

According to how the system collects data, the user feedback is represented either explicitly or implicitly, thus characterizing the recommendation problem as a rating prediction or item prediction, respectively. Explicit feedback refers to the availability of quantified user preferences, generally available in the form of ratings on a numerical scale, e.g., a 1-star to 5-star ranking. The task is then to predict missing values in a matrix of user-item ratings, such as in a regression problem (Vinagre et al., 2014).

However, numerical ratings are not always available due to the necessary systemic adaptations to collect them or because users must provide feedback after their interaction with a given item. Therefore, implicit feedback is more common, such as what happens when a user listens to a song, reads a news article, buys a product, and so forth (Rendle et al., 2012). In the examples above, the feedback is provided in a boolean manner, as either an interaction exists (true) or not (false). Consequently, the objective of implicit feedback-based recommender systems is to predict unknown true values in the boolean interaction matrix, thus characterizing a one-class classification problem (Vinagre et al., 2014).

2.1. Concept drift

An emerging topic in recommender systems is how to learn from potentially infinite sequences of user-item interactions that arrive over time. The building of models that learn incrementally from continuous flows of data is a concern of data stream mining (Aggarwal, 2007; Gama, 2010). Models tailored for data streams must have low computational cost regarding processing time and memory consumption while also tackle concept drifts. Concept drift is a problem that arises when the data distribution changes, thus affecting its underlying patterns (Tsymbal, 2004; Webb et al., 2018). If a model cannot detect and adapt to drifts swiftly, prediction power is put in jeopardy.

In recommender systems, concept drift refers to changes in customer behavior, which reflect changes in their preferences (Jorge et al., 2016). Consequently, recommender systems should be aware that the relationship between users and items is not static. Thus, these are expected to detect and adapt to changes accordingly (Babüroğlu et al., 2021; Gama et al., 2014).

The most efficient way to deal with potentially drifting scenarios is to increment the model as user-item interactions are made available. Recommender systems that are tailored to handle data streams address both the problem of learning drifting behavior and computational complexity issues. In Section 3.2, we discuss incremental algorithms tailored for streaming recommender systems.

2.2. The cold start problem

Most collaborative filtering approaches require a large number of ratings from a user on an item to provide useful recommendations, thus leading to unreliable suggestions due to an initial absence of ratings (Ocepek et al., 2015). Although collaborative filtering is widely followed in recommender systems' implementation, its techniques suffer from sparsity and cold start problems (Wei et al., 2017). Data sparsity occurs when the number of interactions between users and items is much smaller than the total number of user and item combinations. Consequently, the mapping between the latent factors and the rating matrix R becomes even more cumbersome (Shao et al., 2021).

On the other hand, the cold start problem occurs when either a new user or item appears over time. Thus, the recommender model cannot map the user-item interactions accurately as no *a priori* information on either of them is available. In recommender systems, the cold start problem includes three cases: (1) cold start of users (how to recommend items to a user recently entered the system); (2) cold start of items (how to recommend a new item recently introduced into the system to interested users); and (3) cold start of the system (how to realize accurate recommendation in a new system) (Pandey and Rajpoot, 2016; Wei et al., 2017).

Several techniques have been proposed in the literature to solve the cold start problem and deal with data sparseness (Pandey and Rajpoot, 2016; Wei et al., 2017; Silva et al., 2019; Tahmasebi et al., 2020; Bi et al., 2020). Yet, most of these techniques require additional data, such as demographic and social network information. Nonetheless, in real-world scenarios, additional information is often unavailable, and the system has to deal with the provided data to alleviate these problems. A recent approach for addressing cold-start scenarios is using data stream methods, as incremental approaches are continuously learning the new underlying patterns between users and items (Chandramouli et al., 2011; José et al., 2020).

3. Positive-only approaches for recommender systems

In this section, we bring forward existing approaches for positive-only recommender systems. We organize the existing methods into batch and streaming approaches depending on how training takes place.

3.1. Batch approaches

The most common approach for recommender systems with positive-only feedback is Matrix Factorization (MF) (Takács et al., 2009; Yu et al., 2016). MF models have been widely applied to different recommendation scenarios and overcame clustering and neighborhood methods in terms of predictive power and runtime complexity (Vinagre et al., 2014). Thus, we describe traditional MF techniques, including Singular Value Decomposition (SVD) (Sarwar et al., 2000) and Bayesian Personalized Ranking for Matrix Factorization (BPRMF) (Rendle et al., 2012). Furthermore, we also report recent approaches that use neural networks with and without matrix factorization, such as Generalized Matrix Factorization (GMF), Multi-layer Perceptron (MLP), and Neural Matrix Factorization (NeuMF) (He et al., 2017).

3.1.1. Singular value decomposition (SVD)

The Netflix challenge (Funk, 2006) popularized the use of matrix factorization using Singular Value Decomposition (SVD) (Paterek, 2007) or Regularized Singular Value Decomposition (RSVD). SVD factorizes a user-item matrix that represents the interaction between users and items into low-rank matrices.

Matrix factorization characterizes both users and items by vectors of latent factors. The number of latent factors is much smaller than the number of users and items, and the co-occurrence between users and items is the basis for recommendations (Takács et al., 2009).

More formally, matrices $R \in \mathbb{R}^{mxn}, A \in \mathbb{R}^{mxk}$ and $B \in \mathbb{R}^{kxn}$ represent users' ratings to items, users' latent vectors, and items' latent vectors, respectively. The entry $r_{u,i}$ in the u-th row and the i-th column of R is the rating that user u gives to the item i. The u-th row vector p_u of A and i-th column vector q_i of B are user u and item i's latent vectors, respectively (Yu et al., 2016). Given this formulation, we can compute the user u predicted rating for the item i using the dot (scalar) product, as depicted in Eq. 1 (Vinagre et al., 2014).

$$\widehat{R}_{ui} = A_u \cdot B_i^T \tag{1}$$

Training is performed by minimizing the L2-regularized squared error for known values in R and the corresponding predicted ratings as denoted in Eq. 2 (Vinagre et al., 2014).

$$\min_{A,B} \sum_{(u,i)\in D} \left(R_{u,i} - A_u B_i^T\right)^2 + \lambda_A \|A_u\|^2 + \lambda_B \|B_i\|^2 \tag{2}$$

The most common approach for minimizing Eq. 2 is the Stochastic Gradient Descendent (SGD) optimization, in which the algorithm loops over all ratings in the training set. For each given training interaction, the system predicts r_{ui} and computes the associated prediction error computed using Eq. 3 (Koren et al., 2009).

$$err_{ui} = R_{ui} - \widehat{R}_{ui} \tag{3}$$

Next, SGD modifies the parameters by a magnitude proportional to η in the inverse direction of the gradient of the error according to Eq. 4 (Koren et al., 2009), where η is known as step size, or learning rate, and λ is a regularization factor for both users and items latent factors (Vinagre et al., 2014).

$$\begin{aligned} A_u &\leftarrow A_u + \eta(err_{ui} B_i - \lambda A_u) \\ B_i &\leftarrow B_i + \eta(err_{ui} A_u - \lambda B_i) \end{aligned} \tag{4}$$

3.1.2. Bayesian personalized ranking for matrix factorization (BPRMF)

Bayesian Personalized Ranking for Matrix Factorization (BPRMF) is a generic optimization criterion for personalized ranking that is the maximum posterior estimator derived from a Bayesian analysis of the problem (Rendle et al., 2012). The authors provide a learning BPR method based on SGD with bootstrap sampling (Breiman, 1996). In addition to the matrix factorization parameters, for each instance $(u; i)$ in the training set, BPRMF selects a negative item j (an item that the u-th user did not interact with). BPR optimization decomposes triplets in the $(u; i; j)$ format using the difference of the u-th user predictions w.r.t. items i and j, such as depicted in Eq. 5.

$$\widehat{R}_{uij} = \widehat{R}_{ui} - \widehat{R}_{uj} \tag{5}$$

For each triplet $(u; i; j)$, the latent factor vectors for a user u and items i and j are updated using Eq. 6.

$$\Theta \leftarrow \Theta + \eta\left(\frac{e^{-\widehat{R}_{uij}}}{1 + e^{-\widehat{R}_{uij}}} \times \frac{\partial}{\partial_\Theta}\widehat{R}_{uij} - \lambda\Theta\right) \tag{6}$$

where

$$\frac{\partial}{\partial_\Theta}\widehat{R}_{uij} = \begin{cases} (B_i - B_j) & \text{if} \quad \Theta = A_u \\ A_u & \text{if} \quad \Theta = B_i \\ -A_u & \text{if} \quad \Theta = B_j \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

3.1.3. Generalized matrix factorization

The Generalized Matrix Factorization (GMF) model results from an extensive literature investigation of factorization models. It is a particular case of the Neural Collaborative Framework (NCF) (He et al., 2017). As input, GMF receives the one-hot encoded feature vectors v_u^U and v_i^I that describe user u and item i, respectively. Above the input layer are the embedding layers $A_u = A^T v_u^U$ and $B_u = B^T v_i^I$, which are the latent vectors of user and item, respectively. NCF's first layer mapping function is given by Eq. 8, where $\odot$ denotes the element-wise product of user and item latent vectors.

$$\phi(A_u, B_i) = A_u \odot B_i, \tag{8}$$

Each embedding layer (latent vector) is a fully connected layer that projects the sparse representation of users and items into a dense vector. Thus, projecting the vector to the output layer, the rating prediction of the u-th concerning the i-th item is obtained, as denoted in Eq. 9, where a_{out} and h represent the activation function and edge weights of the output layer, respectively.

$$\widehat{R}_{ui} = a_{out}\left(h^T(A_u \odot B_i)\right), \tag{9}$$

Finally, the matrix factorization model is computed using an identity function for a_{out} and enforcing h to being a uniform vector of ones (He et al., 2017).

3.1.4. Multi-layer perceptron (MLP)

Similar to GMF, the multi-layer perceptron (MLP) for recommender systems is also part of the NCF framework (He et al., 2017). The first difference between GMF and MLP resides in the first layer. MLP concatenates the user and item latent features instead of using the element-wise dot product between latent factors as in GMF. Second, the model has hidden layers on the concatenated vector to model the collaborative filtering effect and learn the interaction between A_u and B_i latent features. More formally, Eq. 10 describes the MLP model, where W_x, b_x, and a_x denote the weight matrix, bias vector, and activation function for the x-th layer, respectively.

$$
\begin{aligned}
z_1 &= \phi_1\left(A_u, B_i\right) = \begin{bmatrix} A_u \\ B_i \end{bmatrix}, \\
\phi_2(z_1) &= a_2\left(W_2^T z_1 + b_2\right), \\
&\cdots\cdots \\
\phi_L\left(z_{L-1}\right) &= a_L\left(W_2^T z_L + b_L\right), \\
\widehat{R}_{ui} &= \sigma\left(h^T \phi_L(z_{L-1})\right),
\end{aligned} \tag{10}
$$

3.1.5. Neural matrix factorization

Neural Matrix Factorization (NeuMF) combines GMF and MLP architectures. More specifically, it combines the linear kernel from GMF and the non-linear kernel from MLP. Internally, NeuMF trains GMF and MLP with random initializers until convergence. To provide more flexibility to the combined model, NeuMF allows GMF and MLP to learn separated embeddings and merge them by concatenating their last hidden layer. A formal description for NeuMF is given in Eq. 11 (He et al., 2017), where A_u^G and A_u^M denote the user embedding for GMF and MLP parts, respectively, while similar notations of B_i^G and B_i^M hold for item embeddings.

$$
\begin{aligned}
\phi^{GMF} &= A_u^G \odot B_i^G, \\
\phi^{MLP} &= a_L\left(W_L^T\left(a_{L-1}\left(\ldots a_2\left(W_2^T\begin{bmatrix} A_u^M \\ B_i^M \end{bmatrix} + b_2\right)\ldots\right)\right) + b_L\right), \\
\widehat{R}_{ui} &= \sigma\left(h^T\left(\begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix}\right)\right),
\end{aligned} \tag{11}
$$

3.2. Streaming approaches

In real-world applications, users' feedback is made available continuously. Besides, we have no control over the order in which this data arrives. Let us suppose that new information becomes available in batch mode, such as a new item, a new user, or a new user-item event. In that case, we need to retrain the entire matrix factorization (MF) model using both old and new data. The retraining process may be unfeasible since batch training is computationally expensive (Yu et al., 2016; Wu et al., 2008). Furthermore, there is no guarantee that the past user-item relationships are consistent with those observed in more recent data. Consequently, we should update the recommender systems in a single-pass manner, according to the arrival of user-item interactions, instead of retraining the entire model (Vinagre et al., 2014). As a result, the practical benefits of using single-pass incremental systems encompass computational cost reduction (Wu et al., 2008) and drift adaptation (Matuszyk et al., 2015; Chang et al., 2017).

In this section, we present two techniques designed to work incrementally: Incremental Stochastic Gradient Descendent (ISGD) (Vinagre et al., 2014) and Incremental Bayesian Personalized Ranking for MF (IBPRMF) (Rendle et al., 2012). These methods update latent factor matrices A and B considering the order in which the data becomes available. Besides, these have two fundamental differences between their batch versions, i.e., the learning process is performed on a single data pass (Gaber et al., 2005); and no data shuffling happens so that the natural order in which user-item interactions occur is preserved.

3.2.1. Incremental stochastic gradient descent (ISGD)

The Incremental Stochastic Gradient Descent (ISGD) is designed to deal with positive-only feedback. The interactions between users and items are represented in the boolean matrix R, where $R_{u,i} = 1$ stands for the existence of a relationship between the u-th user and the i-th item, and $R_{u,i} = 0$ depicts the absence of such relation. Therefore, the error is measured as $err_{ui} = \left|1 - \widehat{R}_{ui}\right|$ and the rows in A and B^T are updated using the operations in Eq. 4, adjusting them in the inverse direction of the gradient of the error, by a factor of $0 \leqslant \eta \leqslant 1$ (Vinagre et al., 2014).

One distinct difference between ISGD and its batch counterpart (SVD) regards the order in which the data are analyzed to generate the models. SVD shuffles and performs multiple passes over the data during pre-processing and model creation. On the other hand, ISGD does not perform any data pre-processing and processes data according to their natural arrival order (Vinagre et al., 2014).

3.2.2. Incremental Bayesian personalized ranking for matrix factorization (IBPRMF)

The Incremental Bayesian Personalized Ranking for Matrix Factorization (IBPRMF) updates vectors A_u, B_i and B_j using the operations present in Eqs. 6 and 7 as each user-item interaction of the dataset is made available.

Another significant difference between IBPRMF and BPRMF resides in the set of negative items made available for selection during a user-item interaction model update. In the streaming variant (IBPRMF), the negative item is selected based on items that appeared thus far in the stream and did not interact with the current user.

4. Supermarket dataset with implicit feedback SMDI)

This section describes the Supermarket Dataset with Implicit Feedback (SMDI) used in this case study, broadening data acquisition, pre-processing, and descriptive statistics.

4.1. Data acquisition

The original dataset came from a supermarket that is a customer of HiMarket[1], and it encompasses purchases made between August 1st, 2019, and November 30th, 2019. It contains 737,893 events representing 9531 customers purchasing 7151 items in supermarket transactions carried out on-site since the customer analyzed does not provide delivery services. The events are sorted chronologically and reported in the ⟨user; item; rating; timestamp⟩ form, as illustrated in Fig. 1.

Fig. 2 provides insights on purchases' temporal traits considering different granularities (day, week, and month). In the daily plot provided in Fig. 2a, we observe that the number of items sold over time drastically reduces. The reason is that the beginning of the data timespan matches the rewards club launch and that the interest decreased over time. Furthermore, we observe that the number of purchases increases during weekends (Fig. 2b) and and in the first and last three days of the month (Fig. 2c[2]).

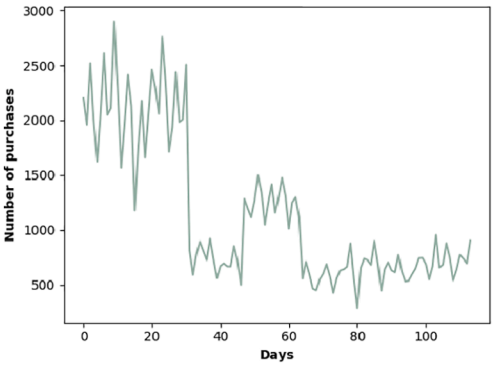
One particular consideration of the SMDI dataset is that it contains repeated events, i.e., users may purchase the same item multiple times. Consequently, Fig. 3a depicts the dataset 'imbalance' as the maximum and minimum amount of user interactions is volatile. On the other hand,

user_id	item_id	rating	timestamp
1078	4175	1	1564628400
2317	6535	1	1564628400
2317	1499	1	1564628400
2317	4175	1	1564628400
2317	123	1	1564628400
...	...	...	...
4901	687	1	1575082800
4901	496	1	1575082800
1026	2827	1	1575082800
1026	6960	1	1575082800
4198	2531	1	1575082800

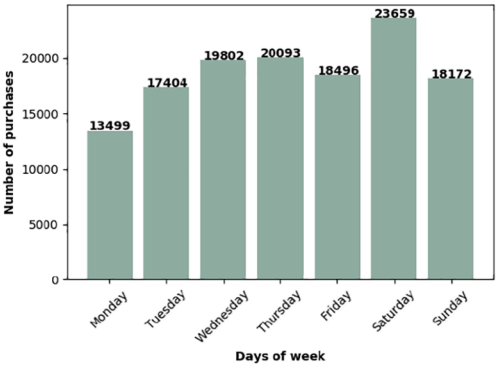
Fig. 1. Dataset representation.

[1] Himarket is a company located at Curitiba, Paraná, Brazil. HiMarket's website is available at http://himarket.club/

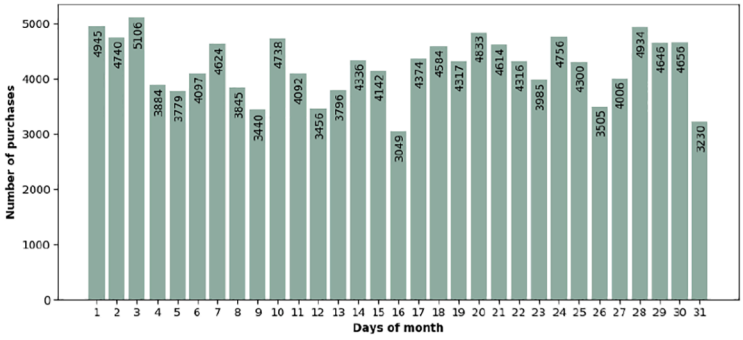
[2] An exception regards the 31st day, as not all months have 31 days.



(a) Daily number of purchases.

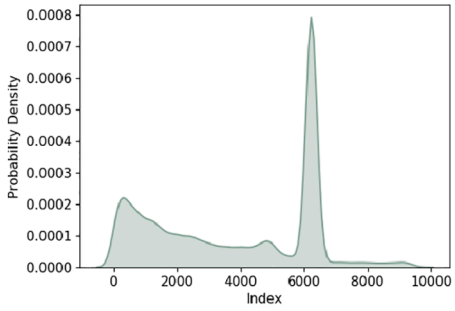


(b) Number of purchases per day of week

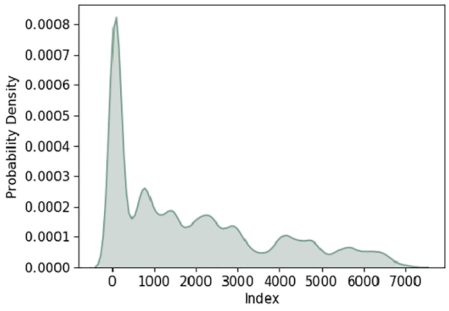


(c) Number of purchases per day of month

Fig. 2. Number of purchases considering different timestamp granularity.



(a) Number of user interactions.



(b) Number of items occurrences.

Fig. 3. Probability density for users (a) and items (b) in the original SMDI dataset.

Fig. 3b shows that some items have a higher number of interactions than others, as these are essential items. Given the characteristics mentioned above, there are two problems related to the supermarket process that contribute to its 'imbalancedness.' First, we observe that cashiers enter their identification number to enable discounts to customers who do not provide their rewards club code when making purchases. Consequently, some users possess an unrealistic number of purchases. Second, users with a large number of events represent merchants in the region close to the supermarket. Thus, the next section presents two approaches to address these problems and reduce the dataset's imbalance characteristic.

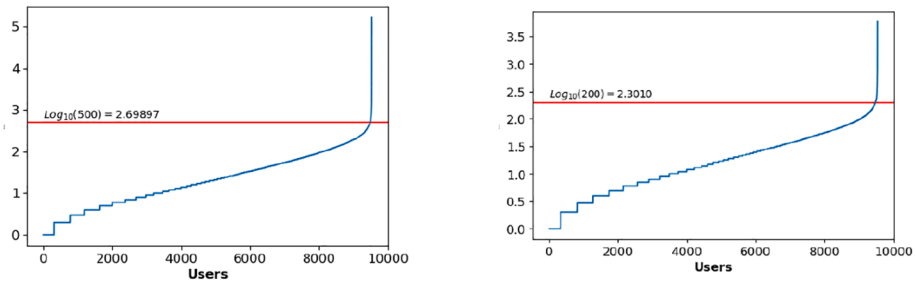
4.2. Dataset pre-processing approaches

This section presents two pre-processing approaches to alleviate the 'imbalancedness' in the number of user events in the SMDI dataset. These strategies account for the total number of events per user and the number of unique events per user, respectively.

4.2.1. Cut-off point by the total number of events per user

The first approach filters the dataset according to the maximum number of events per user. Fig. 4a reports the $\log_{10}$ of the number of events per user. In this strategy, we assume as a cut-off point the region of the curve in which the number of interactions substantially rises, i.e., where the approximate value of $\log_{10}$ is 2.7; thus indicating a maximum number of approximately 500 ($\log_{10}(500) = 2.69897$) interactions per user. We refer to this version of the dataset as `SMDI-500E`.

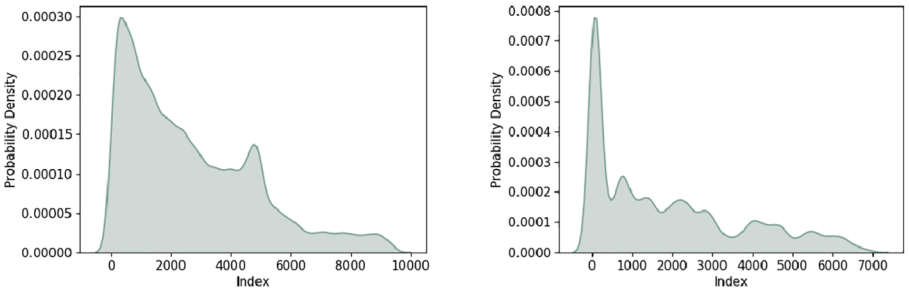
The resulting density distributions obtained for users and items with the removal of such users are depicted in Fig. 5a and b, respectively. Despite the removal of users in this pre-processing approach, the



(a) Distribution of the number of events (N) per user.

(b) Distribution of the number of unique events per user.

Fig. 4. $\log_{10}$ transformation applied to ordered events (N) for each user (a) and unique events per user (b).



(a) Probability distribution of users' occurrences.(b) Probability distribution of items' occurrences.

Fig. 5. Probability distribution for users (a) and items (b) in the `SMDI-500E` dataset.

distribution observed for the number of items (Fig. 5b) does not largely differ from the one observed in the original dataset (Fig. 3b). On the other hand, with the removal of users with more than 500 interactions, the 'imbalancedness' was reduced. In Table 2, we observe the characteristics of the pre-processed dataset. We observe that the maximum number of interactions per user is 500, regardless of whether there are multiple interactions with the same items.

4.2.2. *Cut-off point by the number of unique events per user*

When analyzing the number of unique interactions per user in the original dataset in Fig. 4b and Table 2, we also observe another relevant imbalance in the dataset. More specifically, the number of unique events for cashiers is close to the total number of items available in the dataset and reinforces the need to remove such users.

Following the same rationale for picking a cut-off point, an analysis of Fig. 4b showed that the number of unique events per user rapidly increases with an approximate value of 2.3 for the $\log_{10}$ transformation of the number of unique events, thus indicating the removal of users who interacted with more than 200 ($\log_{10}(200) = 2.3010$) different items in the analyzed period. Consequently, we refer to this dataset as `SMDI-200E` in the remainder of the paper.

The probability distribution for users and items obtained after this pre-processing approach are given in Fig. 6a and b, respectively. Similarly to the previous section's approach, a significant reduction in the number of interactions per user has been observed (Fig. 6a). At the same time, the probability curve for items (Fig. 6b) does not significantly differ from the one observed in the original dataset (Fig. 3b). Table 2 depicts this variant characteristics.

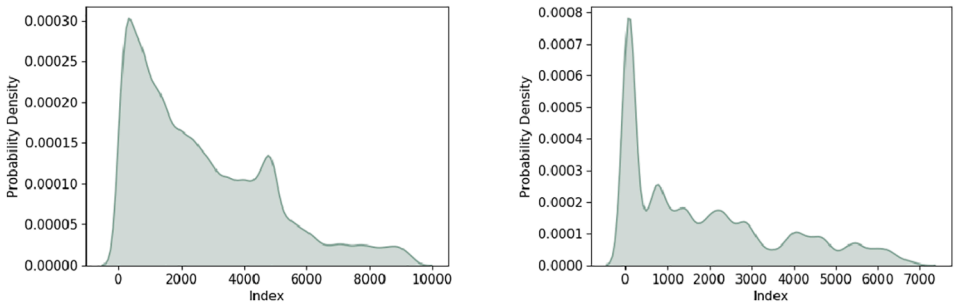
Table 2
Description of the SMDI dataset variants.

Information	Original	SMDI-500E	SMDI-200UE
Events	737,893	448,791	447,391
Unique events	292,943 (39.7%)	268,592 (59.84%)	266,354 (59.84%)
Users	9531	9,480	9472
Items	7141	6933	6924
Avg # of events	77	47	47
Min # of events	1	1	1
Max # of events	166,549	491	775
Sparsity	99.57%	99.59%	99.59%
Avg # of unique events	30	28	28
Min # of unique events	1	1	1
Max # of unique events	5853	270	200

4.3. *Dataset availability and content*

Table 1 overviews the dataset's content, particularly the included files, their respective formats, and pieces of information. The dataset, its pre-processed versions, and the source code to replicate the experiments described in this paper can be downloaded from http://www.ppgia.pucpr.br/jean.barddal/datasets/SMDIDataset.zip. The original events without any pre-processing are in the `SMDI_original.csv` file. The `SMDI-500E.csv` and `SMDI-200UE.csv` files contain the results of pre-processing approaches shown in Section 4.2.1 and 4.2.2, respectively. Users and items information are presented in `SMDI_users.csv` and `SMDI_items.csv`, respectively.

On the item information level, the SMDI dataset includes section and brand identifiers (`section_id` and `brand_id`), reference price (`ref_price`), average, minimum and maximum prices during the period (`avg_price`, `min_price`, and `max_price`), and amount. Since users' personal information is not made available and only purchase data was collected, we calculate, based on the obtained data, the Recency-Frequency-Monetary (`RFM`) score (Weng, 2017). Recency (`R`) is the interval between the current and previous purchase, and thus, the shorter the interval is, the bigger `R` is. Frequency (`F`) indicates the user's number of transactions in a particular period. The bigger the frequency



(a) Number of user interactions. (b) Number of items occurrences.

Fig. 6. Probability density for users (a) and items (b) in the SMDI-200E dataset.

Table 1
Description of the files constituting the SMDI datasets. Bold and underlined contents represent the same information across different files.

File	Format	Content
SMDI_original	csv	**user_id**, item_id, rating, timestamp
SMDI-500E	csv	
SMDI-200UE	csv	
SMDI_users	csv	**user_id**, R, F, M, RFM, R1, F1, M1, RFM1, R2, F2, M2, RFM2, R3, F3, M3, RFM3, R4, F4, M4, RFM4
SMDI_items	csv	item_id, section_id, brand_id, ref_price, avg_price, min_price, max_price, amount

is, the bigger F is. Finally, Monetary (M) refers to all transactions' monetary value in a specific period.

The RFM analysis assigns three different scores related to recency, frequency, and monetary variables to each customer, using a scale from 1 to 5. Therefore, the database is sorted per RFM dimension, and the customer list is divided into five equal segments. The top quintile is assigned a score of 5, and the others receive 4, 3, 2, and 1 (Christy et al., 2018), according to the quintile they belong. The RFM score is then generated by concatenating R, F, and M components, in this specific order. The normalized recency, frequency, and monetary values, calculated monthly (variables R1, F1, M1; R2, F2, M2; and so forth) and the entire period's value are available in the SMDI_users.csv file.

4.4. Descriptive statistics

Table 2 provides statistics from the variants of the SMDI dataset. Regarding the pre-processing methods discussed in Sections 4.2.1 and 4.2.2, we see that both pre-processing approaches obtained close values to the total number of interactions (448791 and 447391), as well as the number of users (9480 and 9472) and items (6933 and 6924). Conversely, the maximum number of events per user in the SMDI-500E dataset is 775, which is much higher than 491 seen in the SMDI-200UE variant. For both pre-processed datasets, the average number of interactions (Avg # of events) per user was the same (47).

Table 2 also presents the statistics for each dataset w.r.t. unique events. Despite the maximum number of unique events per user in the original dataset (5853) being much higher than for pre-processed datasets (270 for both), the average number of unique events per user (Avg # of events) for both is roughly the same (30 for the SMDI_original dataset and 28 for the SMDI-500E and SMDI-200UE datasets).

5. Experimental protocol

This section describes the experimental protocol used to compare batch and streaming algorithms in the SMDI dataset. This experimental protocol is relevant to guarantee that batch and streaming methods are adequately compared and enable identifying concept drifts and cold start problems.

Fig. 7 shows the proposed batch and stream protocols. The dataset was split using the first two months of data for training and the remainder two months for testing. The temporal split makes more sense than a random one because users' interest may change over time (Matuszyk et al., 2015). It is also more realistic as it mimics the data behavior if any of the recommender systems were applied in the real-world (Sidana et al., 2017), thus making a fair comparison between batch and stream learning algorithms.

During the training step, batch and streaming algorithms have significant differences. For batch training, we first shuffle the data and use 20% of it for validation. The validation set is applied to monitor the validation loss, thus allowing early stopping during training. Finally, we use the test set for assessing the recommender system on unseen data.

Regarding streaming models, the first 20% of the training set is used solely for training. The rationale is to allow the streaming recommender system to learn initial parameters, uncover user-item relationships embedded within the latent factors, and output non-random recommendations at the beginning of the experiment. We use the remainder of the training set for testing and incremental training. Data shuffling is not performed as the instances' natural order must be preserved (Vinagre et al., 2014). Next, the test set is used in a test-then-train fashion, meaning that user-item interaction is queried and later used for model update Gama et al. (2013).

We implemented the batch models on top of TensorFlow (Abadi et al., 2016) and streamers using native Python. The source code for all the experiments reported in this paper is made available alongside the dataset in the same link, i.e., http://www.ppgia.pucpr.br/jean.barddal/datasets/SMDIDataset.zip. All experiments were performed on an Intel i7-based computer equipped with 64 GB of RAM, an NVIDIA Titan V with 12 GB of RAM, and an NVIDIA RTX 2070 SUPER with 8 GB of RAM. For the NCF models (GMF, MLP, and NeuMF), instead of only using the positive examples to modeling the relationship between users and items, we randomly sampled four unknown items per positive example to serve as negative ones according to the protocol suggested in He et al. (2017). Considering the paired model BPRMF, it requires a single negative instance per positive interaction during training. Thus, we randomly selected a negative example to balance the positive–negative item pairs. To deal with the negative examples in the NCF models, we used the Binary Cross-Entropy (Log-loss) loss functions, shown in Eq. 12, where $|T|$ is the number of training or validating samples.

$$\text{Log}-\text{loss} = -\frac{1}{|T|}\sum_{u,i\in T}\left(R_{u,i}\log\left(\widehat{R}_{ui}\right) + \left(1-R_{u,i}\right)\times\left(\log\left(1-\widehat{R}_{ui}\right)\right)\right) \quad (12)$$

For the other methods (SVD, BPRMF, ISGD, and IBPRMF), we used Mean Squared Error (MSE) as the loss function, which is depicted in Eq. 13).

$$\text{MSE} = \frac{1}{|T|}\sum_{u,i\in T}\left(R_{u,i}-\widehat{R}_{ui}\right)^2 \quad (13)$$

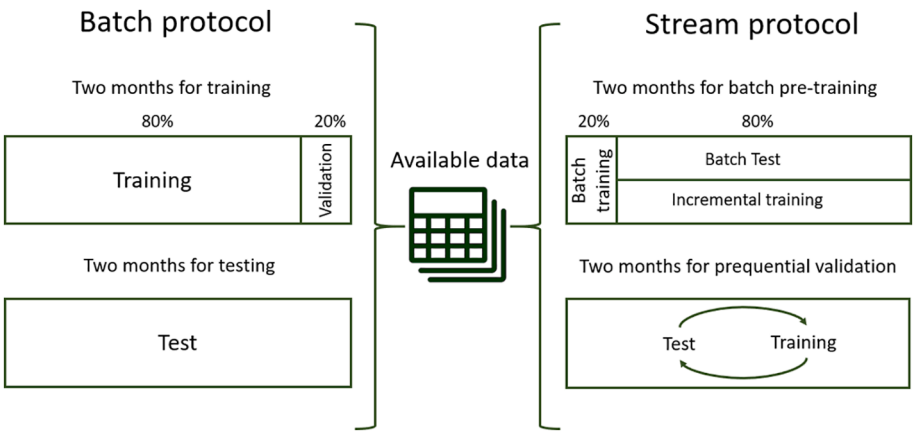


Fig. 7. Batch and stream protocols.

The model parameters were randomly set according to a Gaussian distribution with $\mu = 0$ and a $\rho = 0.01$. Hyper-parameter tuning was performed on a recommender model and dataset basis. We tested the following hyper-parameter values for SVD, BPRMF, ISGD, and IBPRMF: learning rate (learning-rate) $\in$ [0.01, 0.02, 0.05, 0.001, 0.005, 0.0001, 0.0005], regularization rate (reg-rate) $\in$ [0.01, 0.001, 0], latent factors $\in$ [5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100] for SVD, BPRMF, ISGD and IBPRMF. For GMF, MLP, and NeuMF models, we tested the values of latent factors $\in$ [8, 16, 32, 64, 128], and as suggested in the original paper (He et al., 2017), the embedding layers do not have regularization, i.e., reg-rate $= 0$. The number of negative items selected for the NCF models (GMF, MLP and NeuMF) were `neg_items` $\in$ [0, 1, 2, 3, 4, 5, 10, 15, 20]. In Table 3, we show the best parameters selected per model and dataset.

In this analysis, we use the `RECALL@N` metric (Cremonesi et al., 2010) to assess the goodness of the recommendations obtained. The `RECALL@N` metric quantifies how often a recommender suggests a relevant item (hit) amongst unknown items, which are assumed to be irrelevant. In other words, the `RECALL@N` score, computed using Eq. 14, measures the average (across all users) of the proportion of recommended items that appear on the top N items in the ranked list (Yuan et al., 2011), where $|T|$ denotes the number of user-item interactions assessed.

$$\texttt{RECALL@N} = \frac{1}{|T|}\sum_{u,i\in T}(\text{hit@N}(u,i)) \tag{14}$$

Consequently, the `RECALL@N` metric was computed as follows (Matuszyk and Spiliopoulou, 2017): for each instance $\langle u, i\rangle$ in the test set (T), a candidate list of 100 unknown items to user u is selected, and the known (relevant) item i is appended to this candidate list. The candidate items are ranked according to the scores (probability of interaction with the user) obtained from the recommender system and sorted in descending order. For each instance (u, i), $\text{hit@N}(u, i) = 1$ is said to happen when i is ranked amidst the top N items, and $\text{hit@N}(u, i) = 0$, otherwise.

The protocol followed to assess recall has two variants. The first is an approach we refer to as a 'basic evaluator,' which measures the recommender system's using the entire test set. This approach allows the comparison between recommender systems and hypothesis testing. The second approach is the 'window-based evaluator,' which reports the recall over test set chunks. We use a window with a size at every 1% of the test set.

The rationale behind the window-based evaluation is that it allows the assessment of recommender systems over time. This assessment is critical to verify whether the dataset exhibits drifting characteristics and whether streaming models benefit from the incremental updates performed over test data. Therefore, we followed the Prequential test-then-train process (Gama et al., 2013; Jorge et al., 2016) for validating streaming models as depicted in Fig. 8.

Finally, we incorporate hypothesis testing to determine whether one recommender algorithm outperforms others. We followed the protocol reported in Demsar (2006) by combining Friedman (1937) and the Nemenyi post hoc (Nemenyi, 1963) statistical tests. The experimental results are the mean and standard deviation of 30 replications.

Table 3
Parameters tuning for each model and dataset.

Dataset	Model	Opt	Loss	Factors	Reg-rate	Learning-rate
SMDI_original	SVD	SGD	MSE	40	0.01	0.001
	BPRMF	SGD	MSE	30	0.01	0.0005
	GMF	Adam	Log-loss	32	0	0.001
	MLP	Adam	Log-loss	32	0	0.0001
	NeuMF	Adam	Log-loss	32	0	0.0005
	ISGD	SGD	MSE	10	0.01	0.02
	IBPRMF	SGD	MSE	20	0.001	0.001
SMDI-500E	SVD	SGD	MSE	30	0.01	0.001
	BPRMF	SGD	MSE	40	0.01	0.0001
	GMF	Adam	Log-loss	32	0	0.001
	MLP	Adam	Log-loss	32	0	0.001
	NeuMF	Adam	Log-loss	32	0	0.001
	ISGD	SGD	MSE	10	0.01	0.02
	IBPRMF	SGD	MSE	30	0.001	0.05
SMDI-200UE	SVD	SGD	MSE	40	0.01	0.001
	BPRMF	SGD	MSE	80	0.01	0.0005
	GMF	Adam	Log-loss	32	0	0.001
	MLP	Adam	Log-loss	32	0	0.0001
	NeuMF	Adam	Log-loss	32	0	0.0001
	ISGD	SGD	MSE	10	0.01	0.02
	IBPRMF	SGD	MSE	40	0.001	0.05

6. Experimental results and analysis

This section reports the experimental results observed when comparing batch and streaming recommender algorithms applied to the SMDI datasets. We discuss the observations of the two proposed strategies planned in the experimental protocol, as follows: the basic evaluation in Section 6.1 and the window-based evaluation in Section 6.2.

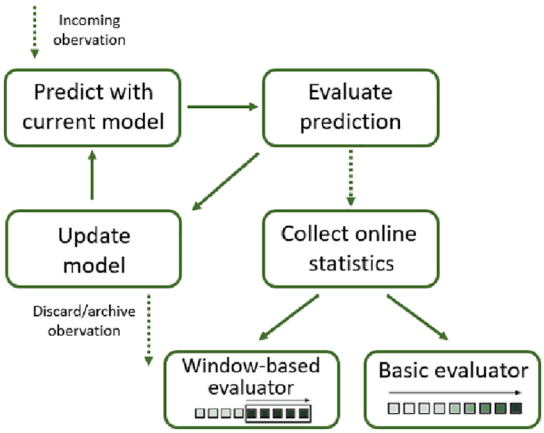


Fig. 8. Prequential validation. Adapted from Jorge et al. (2016).

6.1. Results of the basic evaluation

This section reports the results obtained by batch and streaming models in the SMDI dataset variants. More specifically, we focus on the basic evaluation process described in the proposed experimental protocol (Section 5) in which the recommendation rates are computed over the entire test set. Table 4 shows the general performance obtained.

According to the dataset variants' results, we observe differences in the tested recommender models' behavior. Considering the original dataset results, we verify that there is no clear indication of whether batch or streaming models outperform others. We report the statistical test results obtained by combining Friedman and Nemenyi tests assuming a 95% confidence level throughout our discussion. These differences are also reported in Table 4, where recall values are marked with letters ($a \succ b \succ c \succ d \succ e \succ f$) that depict and group the goodness of their results.

In this table, we also report the results for PopTop (Cremonesi et al., 2010), a non-machine learning approach for assessing recommender systems. PopTop consists of recommending items with the best degree of success among all users instead of modeling the user-item relationship using a machine learning method. Consequently, we observe that PopTop is outperformed by all methods regardless of the recall metric analyzed. This observation depicts that both batch and streaming algorithms model the users' behavior in supermarket purchases adequately and surpass a naive baseline.

When comparing the batch models, we observe that two neural network models, i.e., MLP and NeuMF, outperform GMF, BPFMF, and SVD in most datasets and recall values. These results indicate that the neural networks are suitable to captura complex interactions between users and items. In all datasets, MLP and NeuMF models do not depict a significant difference between each other. Even though NeuMF is a combination of MLP and GMF, GMF alone did not result in better performance as the recall rates observed are not significantly higher than those surveyed for MLP alone.

Regarding the streaming recommender models' results (highlighted in Table 4), ISGD outperformed their batch version (SVD) concerning the `RECALL@10` values in all datasets, thus showing a statistically significant difference with a 95% confidence level. The improvement is more significant in the pre-processed datasets than in the original one. We observe an increase of 2.8% for `RECALL@10` in both `SMDI-500E` and `SMDI-200UE`, in contrast to 0.5% in the original dataset. The discrepancy between streaming algorithms and the corresponding batch counterpart depicts the importance of constantly updating the recommender system as new data becomes available. This claim is further backed up as ISGD obtained superior results when compared to MLP and NeuMF in the pre-processed variants considering `RECALL@1`, `RECALL@5`, and `RECALL@10`.

Table 4
Recall values obtained by the recommendation methods in each tested dataset. The shaded area comprises the results of the data stream algorithms.

Model	RECALL@1	RECALL@5	RECALL@10	RECALL@20
		SMDI_original.csv		
PopTop	0.041_{d}	0.102_{d}	0.150_{e}	0.215_{f}
SVD	**0.325** ± **0.0021**$_{a}$	**0.565** ± **0.0012**$_{a}$	0.669± 0.0011_{b}	0.779 ± $0.0009_{b,c}$
BPRMF	0.242 ± $0.0013_{b,c}$	0.410 ± 0.0018_{c}	0.484± 0.0019_{d}	0.564 ± $0.0014_{d,e}$
GMF	0.218 ± 0.0023_{c}	0.376 ± 0.0037_{c}	0.454± 0.0048_{d}	0.542 ± 0.0055_{e}
MLP	**0.324** ± **0.0032**$_{a}$	0.559 ± 0.0086_{b}	0.665± $0.0096_{b,c}$	**0.790** ± **0.0116**$_{a,b}$
NeuMF	**0.322** ± **0.0125**$_{a}$	0.559 ± $0.0243_{a,b}$	0.664± $0.0257_{b,c}$	**0.790** ± **0.0268**$_{a}$
ISGD	0.298 ± 0.0023_{b}	0.562 ± 0.0012_{b}	**0.674**± **0.0027**$_{a}$	**0.782** ± **0.0020**$_{a,b}$
IBPRMF	**0.324** ± **0.0144**$_{a}$	0.559 ± 0.0075_{b}	0.662± $0.0046_{c,d}$	0.766 ± $0.0038_{c,d}$
		SMDI-500E.csv		
PopTop	0.042_{f}	0.102_{f}	0.150_{f}	0.216_{e}
SVD	0.299 ± 0.0019_{c}	0.543 ± $0.0010_{b,c}$	0.648± 0.0011_{c}	0.757 ± $0.0015_{c,d}$
BPRMF	0.212 ± $0.0031_{d,e}$	0.382 ± $0.0046_{d,e}$	0.457± $0.0050_{d,e}$	0.538 ± 0.0057_{d}
GMF	0.197 ± 0.0025_{e}	0.371 ± 0.0038_{e}	0.454± 0.0041_{e}	0.540 ± 0.0046_{d}
MLP	0.299 ± $0.0108_{c,d}$	0.539 ± $0.0020_{c,d}$	0.645± $0.0016_{c,d}$	**0.779** ± **0.0019**$_{a,b}$
NeuMF	0.293 ± $0.0220_{b,c}$	0.543 ± $0.0057_{b,c}$	0.650± $0.0027_{b,c}$	**0.782** ± **0.0030**$_{a}$
ISGD	**0.317** ± **0.0008**$_{a,b}$	**0.571** ± **0.0004**$_{a}$	**0.676**± **0.0004**$_{a}$	**0.782** ± **0.0005**$_{a}$
IBPRMF	**0.322** ± **0.0007**$_{a}$	**0.565** ± **0.0006**$_{a,b}$	**0.667**± **0.0006**$_{a,b}$	0.772 ± $0.0006_{b,c}$
		SMDI-200UE.csv		
PopTop	0.042_{e}	0.102_{e}	0.150_{e}	0.216_{e}
SVD	0.299 ± 0.0023_{b}	0.544 ± $0.0012_{b,c}$	0.648± 0.0011_{c}	0.757 ± $0.0016_{c,d}$
BPRMF	0.213 ± 0.0028_{c}	0.384 ± $0.0029_{c,d}$	0.459± $0.0031_{c,d}$	0.539 ± 0.0040_{d}
GMF	0.198 ± 0.0026_{c}	0.371 ± 0.0031_{d}	0.453± 0.0036_{d}	0.539 ± 0.0044_{d}
MLP	0.285 ± 0.0586_{b}	0.544 ± 0.0072_{b}	0.666± 0.0029_{b}	**0.785** ± **0.0013**$_{a}$
NeuMF	0.292 ± 0.0235_{b}	0.542 ± 0.0140_{b}	0.667± 0.0035_{b}	**0.784** ± **0.0011**$_{a}$
ISGD	**0.316** ± **0.0009**$_{a}$	**0.570** ± **0.0005**$_{a}$	**0.676**± **0.0005**$_{a}$	**0.782** ± **0.0005**$_{a,b}$
IBPRMF	**0.322** ± **0.0006**$_{a}$	**0.565** ± **0.0008**$_{a}$	0.667± 0.0007_{b}	0.772 ± $0.0007_{b,c}$

It is also noteworthy the analysis between BPR batch and streaming variants, especially in the pre-processed datasets. For instance, IBPRMF increased the `RECALL@10` values up to 21% compared to the batch model BPRMF, while ISGD improved SVD rates by 2.8%. Extending the analysis, the `RECALL@10` and `RECALL@20` results obtained in the original dataset show that IBPRMF had performance decreases, yet, it was not statistically significant. These decreases were due to the volatility of the recall rates observed, as IBPRMF did not converge in all experiment runs.

Overall, we observe that streaming models performed competitive results in all the datasets, except the ISGD model in the original dataset when `RECALL@1` is assessed. In this specific scenario, we observe that IBPRMF is a formidable contender to match traditional matrix factorization techniques (SVD) and even more complex approaches that rely on neural networks (MLP and NeuMF). Summing up, we see that in smaller N values, i.e., $N \in [1, 5, 10]$, either ISGD or IBPRMF overcome

batch models in the SMDI-500E and SMDI-200UE variants and that MLP and NeuMF achieve superior results in RECALL@20. Comparing the results acquired for both recommendation approaches, i.e., batch and streaming, we observe that streaming models resulted in improvements of 2.8% in recall values. In the next section, we further analyze these results from a different perspective. More specifically, we focus on the performance assessment that takes place over time, thus allowing a more fine-grained analysis on why streaming recommender systems exhibit the behavior mentioned above and provide evidence of the existence of concept drifts and cold start in the dataset assessed.

6.2. Results of the window-based evaluation

In this section, we report the recall rates using the window-based process. Fig. 9 shows the results obtained with the assessment taking place at every 1% of the test set. In the same figure, we also report the cumulative number of users and items to verify the cold start problem's impact.

In real-world datasets, the assessment of user-item interactions over time may uncover concept drifts, such as the launch of a product that reduces the popularity of previous versions of the same product or its competitors. Consequently, users' interests and preferences may drift over time, resulting in concept drifts that should be targeted by adaptive recommender systems (Matuszyk et al., 2015; Chang et al., 2017). In Fig. 9, we observe recurrent fluctuations in the RECALL@10 rates, which represent concept drifts. This observation is corroborated by the recall rates obtained by PopTop, thus depicting that as new user-item interactions occur, the overall behavior in the dataset also changes. Such changes are weekly recurring drifts (Gama et al., 2014), thus meaning that the relationship between users and items changes over the week, but it repeats itself across weeks. Recurring concepts are expected in supermarket scenarios as specific sales are repeated along weeks, days of the month, or even months of the year.

In this analysis, we observe that the streaming recommender models, i.e., ISGD and IBPRMF, allowed significant parameter adjustments over time that induced better performance when compared to other models, especially in the pre-processed dataset variants.

Another relevant aspect observed in Fig. 9 regards the performance decrease observed after the processing of 50 thousand interactions. This decrease matches the behavior change of the cumulative number of users in the dataset, thus culminating in a cold start problem. However, even though we notice an abrupt increase in the number of users in all datasets, most algorithms recover from the cold start and maintain good performance as new instances appear, except for the BPRMF and GMF methods. These results show that (i) streaming models, despite built on matrix factorization, recover from cold start issues swiftly and that (ii) recommender models based on neural networks exhibit interesting behavior in cold-start scenarios even though they are not continuously updated. The reason behind this behavior is related to the internal learning process of neural network-based recommender systems, where the user-item interactions result in higher-order embeddings that better generalize the underlying behavior between users and items when compared to traditional matrix factorization. Consequently, given the neural networks' generalization ability, the recommender models extract unseen patterns in user-item interactions and provide useful suggestions in cold-start scenarios.

On the other hand, when we analyze the cumulative number of items, we observe that the increase is gradual. Comparing the behavior between the original and pre-processed variants, we also observe that the latter datasets' increase is slightly faster. This behavior explains why the performance of streaming and neural batch approaches observed in Table 4 in the original dataset is similar. The streaming models are

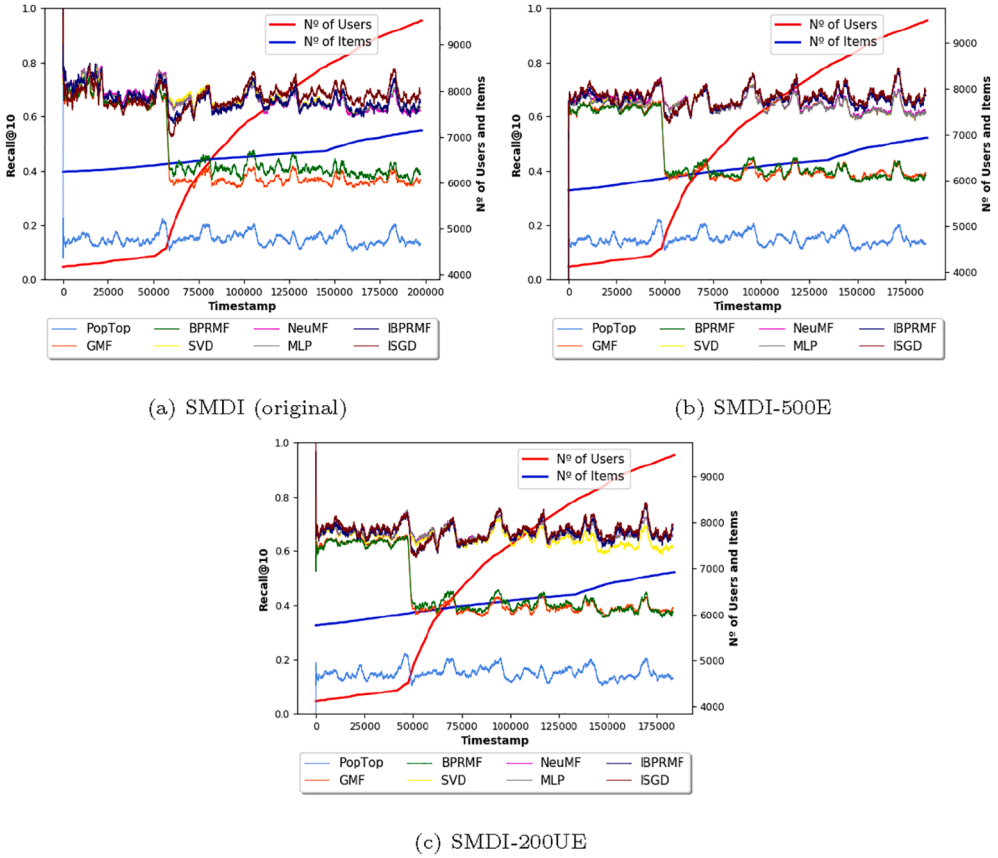


Fig. 9. Moving averages of RECALL@10 values in the test stage, when using a sliding window with size 2000; a) shows the plot evolution obtained in the original dataset; b) pre-processed SMDI-500E dataset; and c) pre-processed SMDI-200UE dataset.

unable to take advantage of noticeable changes in data behavior. In contrast, in the pre-processed datasets where these changes are abrupt, the streaming models can adapt their parameters and achieve higher recall values.

Finally, in both evaluation phases, the streaming methods were more efficient in almost all presented recommendation scenarios. ISGD and IBPRMF outperform their batch versions SVD and BPRMF and obtained better results than observed in the neural network approaches. Considering the concept drift and cold start problems, we verify that the incremental ability of the streaming recommender models improved model prediction accuracy.

7. Conclusion

This paper analyzed batch and stream learning algorithms concerning concept drifts and the cold start problem. As a by-product of this work, we made publicly available a new collaborative filtering supermarket dataset, alongside two pre-processed variants. As a result of this analysis, we observed that streaming recommender systems significantly overcome batch approaches. Thus, more effort should be put into tailoring techniques at the intersection of data streams and recommender systems. For instance, streaming recommender systems were especially beneficial with the occurrence of the cold start issue and overcame complex neural network approaches in weekly recurrent concept drifts with statistical significance.

As future works, we envision scaling up the data acquisition process. A larger data timespan would be beneficial for concept drift analysis across multiple years, yet, such data were unavailable for academic purposes. We also plan to account for the item content such as price, section, and brand that were unused in the current analysis. Thus, an adaptive content-based method or even its combination in an adaptive hybrid approach are envisioned. We also plan to combine explicit drift detection on matrix factorization and neural models in terms of recommendation techniques. Finally, another envisioned approach is to apply content-based filtering techniques in cold start cases, i.e., when a new customer or item appears in the dataset.

CRediT authorship contribution statement

Antônio David Viniski: Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Visualization. **Jean Paul Barddal:** Conceptualization, Methodology, Validation, Formal analysis, Writing - original draft, Visualization, Supervision, Project administration. **Alceu Souza Britto Jr.:** Conceptualization, Writing - review & editing, Supervision, Project administration. **Fabrício Enembreck:** Conceptualization, Writing - review & editing. **Humberto Vinicius Aparecido de Campos:** Resources, Funding acquisition.

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.

Acknowledgements

We would like to thank the Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq (Grant #142195/2019-7) for financing this research and the HiMarket company for the financial support and making the data analyzed in this work available. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P.A., Vasudevan, V., Warden, P., Wicke, M., Yu, Y. & Zheng, X. (2016). Tensorflow: A system for large-scale machine learning. In K. Keeton & T. Roscoe (Eds.), 12th USENIX symposium on operating systems design and implementation, OSDI 2016, Savannah, GA, USA, November 2–4, 2016 (pp. 265–283). USENIX Association.

Aggarwal, C. C. (Ed.) (2007). Data streams – models and algorithms. Volume 31 of advances in database systems. Springer.

Babüroğlu, E. S., Durmuşoğlu, A., & Dereli, T. (2021). Novel hybrid pair recommendations based on a large-scale comparative study of concept drift detection. *Expert Systems with Applications, 163*, Article 113786.

Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). Research-paper recommender systems: A literature survey. *International Journal on Digital Libraries, 17*, 305–338.

Bi, Y., Song, L., Yao, M., Wu, Z., Wang, J. & Xiao, J. (2020). DCDIR: A deep cross-domain recommendation system for cold start users in insurance domain. In J. Huang, Y. Chang, X. Cheng, J. Kamps, V. Murdock, J. Wen & Y. Liu (Eds.), Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, SIGIR 2020, Virtual Event, China, July 25–30, 2020 (pp. 1661–1664). ACM.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems, 46*, 109–132.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*, 123–140.

Chandramouli, B., Levandoski, J. J., Eldawy, A., & Mokbel, M. F. (2011). Streamrec: A real-time recommender system. In *Proceedings of the 2011 ACM SIGMOD international conference on management of data SIGMOD, '11* pp. 1243–1246). New York, NY, USA: Association for Computing Machinery.

Chang, S., Zhang, Y., Tang, J., Yin, D., Chang, Y., Hasegawa-Johnson, M. A., & Huang, T. S. (2017). Streaming recommender systems. In R. Barrett, R. Cummings, E. Agichtein, & E. Gabrilovich (Eds.), *Proceedings of the 26th international conference on world wide web, WWW 2017, Perth, Australia, April 3–7, 2017* (pp. 381–389). ACM.

Christy, A. J., Umamakeswari, A., Priyatharsini, L. & Neyaa, A. (2018). Rfm ranking–an effective approach to customer segmentation. Journal of King Saud University-Computer and Information Sciences.

Cremonesi, P., Koren, Y., & Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In X. Amatriain, M. Torrens, P. Resnick, & M. Zanker (Eds.), *Proceedings of the 2010 ACM conference on recommender systems, RecSys 2010, Barcelona, Spain, September 26–30, 2010* (pp. 39–46). ACM.

Demsar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association, 32*, 675–701.

Funk, S. (2006). Netflix update: Try this at home.

Gaber, M. M., Zaslavsky, A. B., & Krishnaswamy, S. (2005). Mining data streams: A review. *SIGMOD Record, 34*, 18–26.

Gama, J. (2010). *Knowledge discovery from data streams*. CRC Press.

Gama, J., Sebastião, R., & Rodrigues, P. P. (2013). On evaluating stream learning algorithms. *Machine Learning, 90*, 317–346.

Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys, 46*, 44:1–44:37.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. (2017). Neural collaborative filtering. In R. Barrett, R. Cummings, E. Agichtein, & E. Gabrilovich (Eds.), *Proceedings of the 26th international conference on world wide web, WWW 2017, Perth, Australia, April 3–7, 2017* (pp. 173–182). ACM.

Jorge, A. M., Vinagre, J., Domingues, M. A., Gama, J., Soares, C., Matuszyk, P. & Spiliopoulou, M. (2016). Scalable online top-n recommender systems. In D. Bridge, & H. Stuckenschmidt (Eds.), E-commerce and web technologies – 17th international conference, EC-Web 2016, Porto, Portugal, September 5–8, 2016, Revised Selected Papers (pp. 3–20). Volume 278 of Lecture Notes in Business Information Processing.

José, E. F., Enembreck, F., & Barddal, J. P. (2020). Adadrift: An adaptive learning technique for long-history stream-based recommender systems. In *Proceedings of IEEE systems*. IEEE. and Cybernetics 2020 (IEEE SMC 2020).

Koren, Y., Bell, R. M., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *IEEE Computer, 42*, 30–37.

Matuszyk, P., & Spiliopoulou, M. (2017). Stream-based semi-supervised learning for recommender systems. *Machine Learning, 106*, 771–798.

Matuszyk, P., Vinagre, J., Spiliopoulou, M., Jorge, A. M., & Gama, J. (2015). Forgetting methods for incremental matrix factorization in recommender systems. In R. L. Wainwright, J. M. Corchado, A. Bechini, & J. Hong (Eds.), *Proceedings of the 30th annual ACM symposium on applied computing* (pp. 947–953). ACM.

Nassar, N., Jafar, A. & Rahhal, Y. (2020). A novel deep multi-criteria collaborative filtering model for recommendation system. Knowledge Based Systems, 187.

Nemenyi, P. B. (1963). Distribution-free multiple comparisons. PhD thesis Princeton University.

Ocepek, U., Rugelj, J., & Bosnic, Z. (2015). Improving matrix factorization recommendations for examples in cold start. *Expert Systems with Applications, 42*, 6784–6794.

Pandey, A. K., & Rajpoot, D. S. (2016). Resolving cold start problem in recommendation system using demographic approach. In *2016 International conference on signal processing and communication (ICSC)* (pp. 213–218). IEEE.

Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. In Proceedings of KDD cup and workshop (Vol. 2007, pp. 5–8).

Portugal, I., Alencar, P. S. C., & Cowan, D. D. (2018). The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications, 97*, 205–227.

Rendle, S., Freudenthaler, C., Gantner, Z. & Schmidt-Thieme, L. (2012). BPR: Bayesian personalized ranking from implicit feedback. CoRR, abs/1205.2618.

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), *Recommender systems handbook* (pp. 1–35). Springer.

Sarwar, B., Karypis, G., Konstan, J. & Riedl, J. (2000). Application of dimensionality reduction in recommender system-a case study. Technical Report Minnesota Univ Minneapolis Dept of Computer Science.

Shao, B., Li, X., & Bian, G. (2021). A survey of research hotspots and frontier trends of recommendation systems from the perspective of knowledge graph. *Expert Systems with Applications, 165*, Article 113764.

Sidana, S., Laclau, C., Amini, M., Vandelle, G., & Bois-Crettez, A. (2017). In N. Kando, T. Sakai, H. Joho, H. Li, A. P. de Vries, & R. W. White (Eds.), *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval, Shinjuku, Tokyo, Japan, August 7–11, 2017* (pp. 1245–1248).

Silva, N., Carvalho, D., Pereira, A. C. M., Mourão, F., & da Rocha, L. C. (2019). The pure cold-start problem: A deep study about how to conquer first-time users in recommendations domains. *Information Systems, 80*, 1–12.

Tahmasebi, F., Meghdadi, M., Ahmadian, S. & Valiallahi, K. (2020). A hybrid recommendation system based on profile expansion technique to alleviate cold start problem. Multimedia Tools and Applications, (pp. 1–16).

Takács, G., Pilászy, I., Németh, B., & Tikk, D. (2009). Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research, 10*, 623–656.

Tsymbal, A. (2004). The problem of concept drift: Definitions and related work. *Computer Science Department, Trinity College Dublin, 106*, 58.

Vinagre, J., Jorge, A. M. & Gama, J. (2014). Fast incremental matrix factorization for recommendation with positive-only feedback. In V. Dimitrova, T. Kuflik, D. Chin, F. Ricci, P. Dolog & G. Houben (Eds.), User modeling, adaptation, and personalization – 22nd international conference, UMAP 2014, Aalborg, Denmark, July 7–11, 2014. Proceedings (pp. 459–470). Springer Volume 8538 of Lecture Notes in Computer Science.

Webb, G. I., Lee, L. K., Goethals, B., & Petitjean, F. (2018). Analyzing concept drift and shift from sample data. *Data Mining and Knowledge Discovery, 32*, 1179–1199.

Wei, J., He, J., Chen, K., Zhou, Y., & Tang, Z. (2017). Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications, 69*, 29–39.

Wei, J., He, J., Chen, K., Zhou, Y., & Tang, Z. (2017). Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications, 69*, 29–39.

Weng, C. (2017). Revenue prediction by mining frequent itemsets with customer analysis. *Engineering Applications of Artificial Intelligence, 63*, 85–97.

Wu, H., Wang, Y., & Cheng, X. (2008). Incremental probabilistic latent semantic analysis for automatic question recommendation. In P. Pu, D. G. Bridge, B. Mobasher, & F. Ricci (Eds.), *Proceedings of the 2008 ACM conference on recommender systems, RecSys 2008, Lausanne, Switzerland, October 23–25, 2008* (pp. 99–106). ACM.

Yin, R., Li, K., Zhang, G. & Lu, J. (2019). A deeper graph neural network for recommender systems. Knowledge Based Systems, 185.

Yu, T., Mengshoel, O.J., Jude, A., Feller, E., Forgeat, J. & Radia, N. (2016). Incremental learning for matrix factorization in recommender systems. In J. Joshi, G. Karypis, L. Liu, X. Hu, R. Ak, Y. Xia, W. Xu, A. Sato, S. Rachuri, L. H. Ungar, P. S. Yu, R. Govindaraju & T. Suzumura (Eds.), 2016 IEEE international conference on big data, BigData 2016, Washington DC, USA, December 5–8, 2016 (pp. 1056–1063). IEEE Computer Society.

Yuan, Q., Chen, L., & Zhao, S. (2011). Factorization vs. regularization: Fusing heterogeneous social relationships in top-n recommendation. In B. Mobasher, R. D. Burke, D. Jannach, & G. Adomavicius (Eds.), *Proceedings of the 2011 ACM conference on recommender systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011* (pp. 245–252). ACM.

Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys, 52*, 5:1–5:38.