Incremental multi-dimensional recommender systems: co-factorization vs tensors

Miguel Sozinho Ramalho m.ramalho@fe.up.pt
LIAAD - INESCTEC, Porto, Portugal
João Vinagre jnsilva@inesctec.pt
LIAAD - INESCTEC, Porto, Portugal
FCUP - University of Porto, Portugal
Alípio Mário Jorge amjorge@fc.up.pt
LIAAD - INESCTEC, Porto, Portugal
FCUP - University of Porto, Portugal
Rafaela Bastos rafaela.bastos@hostelworld.com
Hostelworld Group, Porto, Portugal

Editors: João Vinagre, Alípio Mário Jorge, Albert Bifet and Marie Al-Ghossein

Abstract

The present paper sets a milestone on incremental recommender systems approaches by comparing several state-of-the-art algorithms with two different mathematical foundations - matrix and tensor factorization. Traditional Pairwise Interaction Tensor Factorization is revisited and converted into a scalable and incremental option that yields the best predictive power. A novel tensor inspired approach is described. Finally, experiments compare contextless vs context-aware scenarios, the impact of noise on the algorithms, discrepancies between time complexity and execution times, and are run on five different datasets from three different recommendation areas - music, gross retail and garment. Relevant conclusions are drawn that aim to help choosing the most appropriate algorithm to use when faced with a novel recommender tasks.

Keywords: Recommender Systems, Matrix Factorization, Matrix Co-Factorization, Tensor Factorization, Incremental Learning, Data Streams

1. Introduction

Recommender Systems (RS) are the set of tools and algorithms, used in a variety of contexts and scales, that recommend items to users. Items can be anything that a company has to offer, from services to products and entities. The ultimate goal of RS is to recommend relevant items, be that from a commercial or pleasure point-of-view.

Ever since their inception in the mid 90s (Venkatesan and Thangadurai, 2016), these tools have been improving in terms of efficiency, predictive ability, processing capacity and in the kind of information used to better describe users, items and the relations between them. The most typical approaches are:

- Content-based: using item's attributes to filter relevant recommendations

- Usage-based: using activity information and/or ratings to implicitly characterize users and items and infer user preferences
- Hybrid: a combination of both strategies to increase the recommendation power

With respect to usage-based approaches, often referred to as Collaborative Filtering (CF), latent factor models have been gaining popularity following the famous Netflix prize competition (Bell and Koren, 2007). These models work by reducing high dimensional data to factors that represent implicit information in past user-item interactions. More, those models range from simple Matrix Factorization (MF) that uses only user-item interactions (Li et al., 2014), to more complex models such as Matrix Co-Factorization (MCF) and Tensor Factorization (TF) that expand those interactions to include relevant user and item attributes and also interaction-specific context.

The distinction between attribute and context is represented in Figure 1. Attributes are usually static (Woerndl and Schlichter, 2007) properties of either users or items (Rendle et al., 2011) - consider, for instance, age, gender, price, color, category as examples of both user and item attributes. Whereas context is interaction-specific (Rendle et al., 2011) and has a more dynamic nature (Woerndl and Schlichter, 2007) - consider, for instance, location information, user mood and weather.

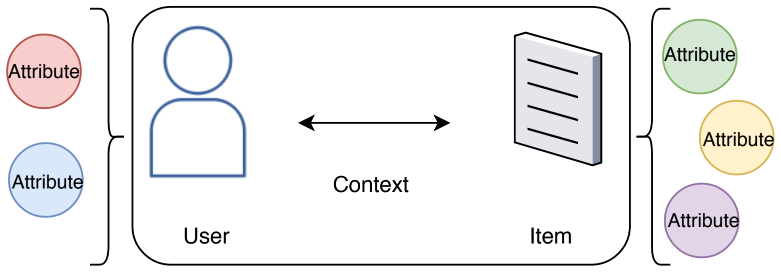

Figure 1: Context and attributes of RS

More recently, the pragmatic ability to process large amounts of data produced at high speed has reached an unprecedented level. As such, real-world RS should be able to handle these challenges and remain informative and efficient. Moreover, recommendation data is usually sparse, as a consequence of a large number of users and items, such that representing data as matrices or tensors becomes unfeasible from a memory standpoint. One of the modeling approaches that best answers all of the above challenges is that of incremental learning.

In essence, incremental approaches use each data instance for learning at the time it is first seen without the need to iterate over all the past instances. These approaches have been implemented by big tech companies and there are plenty of software solutions directly aimed at processing data as streams instead of batches or whole files.

In this paper, we focus on the comparison between matrix and tensor factorization approaches to the item recommendation problem using additional information for different application contexts: music platforms, gross retail sales and garment sales. We compare incremental implementations of the MF and MCF models to a TF technique that has been overlooked in terms of its applications to incremental algorithms. Efforts were also put into the development of a new incremental tensor factorization technique that led to poor results, its negative results are also reported, nonetheless, so that future work on similar techniques can take them into account.

The remainder of this work is structured as follows. In Section 2, we present and explain the inner workings of the incremental factorization models under scrutiny including the attempts at a new incremental tensor factorization algorithm and the adaptation of a non-incremental algorithm into a novel incremental version. Section 3 describes how these incremental algorithms are evaluated and compared. In Section 4, we present the selected datasets, how each of the different models performs and draw conclusions from these results. Finally, Section 5 focuses on the takeaways from the present analysis and on what the future holds for this research topic.

2. Incremental Factorization Models

Factorization models decompose large matrices or tensors into lower dimensional ones, making them easier to manipulate. This reduces memory requirements and computational time while allowing to cope with the typical sparsity of natural user-item interactions. The decomposition process extracts latent factors from the original data, which can be useful by themselves or along with other data mining techniques – see, for example: (Lee and Seung, 1999).

RS data can be represented using arrays, and hence modeled through factorization models. The relations between user-item, user-attribute and item-attribute are represented in matrices. On the other hand, tensors are used when we want to introduce context into the modeling. As mentioned before, context can be the location of the user-item interaction, in this scenario we would have a user-item matrix for each location. Considering the user-item-context case, we would have a 3-dimensional tensor. As context is part of the interaction itself (as it happens), it is considered as another dimension of the array. However, tensors are not limited to context data and can be used with user or item attributes. We actually take advantage of this in our experiments in Section 4.

In the next subsection, details are given about each factorization model that can be used in incremental RS. The notation used is presented in Table 1. Furthermore, Algorithm 1 contains the typical structure for processing incremental events, train the models and evaluate them, as such it will stay relevant for the presented algorithms considering that each algorithm has its own update steps and although these differ, they fit into the same template section. Furthermore, algorithms that are not context aware should skip the if-condition that regards context in the events, denoted with *.

2.1. Matrix Factorization

Matrix factorization (MF), as the simplest approach, uses only user-item interactions data and thereby it represents an easily understandable and implementable algorithm. The interaction is represented by a matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$. This matrix is also known as *feedback matrix*, with m users and n items. Each value r_{ui} of $\mathbf{R}$ is 1 if user u interacts with item i and 0 otherwise. The MF model decomposes $\mathbf{R}$ into two low-rank matrices $\mathbf{A} \in \mathbb{R}^{m \times k}$ and $\mathbf{B} \in \mathbb{R}^{n \times k}$ that cover a common k-dimensional latent space: $\mathbf{R} \approx \mathbf{A}\mathbf{B}^{\mathrm{T}}$. Matrix $\mathbf{A}$ spans the user space, while $\mathbf{B}$ spans the item space. After the decomposition is done, the predicted

Notation	Definition
$X, \mathbf{X}, \mathbf{x}, x$	Tensor, matrix, vector, scalar
$\|\mathbf{x}\|$	Vector norm (ℓ^2-norm)
$\|\|\mathbf{X}\|\|_F$	Frobenius norm
$\#(set)$	Cardinality
$*$	Element-wise multiplication
$\circ$	Outer product

Table 1: Table of symbols

```
Input  : $k$ (iterations), $\lambda$, $\eta$
Output: $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$
for $(u,i,f) \in D$ do
    if $u \notin \texttt{rows}(\mathbf{A})$ then
        $\mathbf{a}_u \sim N(0, 0.1)$
    end
    if $i \notin \texttt{rows}(\mathbf{B})$ then
        $\mathbf{b}_i \sim N(0, 0.1)$
    end
    *For context-aware algorithms also check:
    if $f \notin \texttt{rows}(\mathbf{C})$ then
        $\mathbf{c}_f \sim N(0, 0.1)$
    end
    for $t \leftarrow 1$ to $k$ do
        Perform incremental updates specific to each algorithm
    end
end
```

Algorithm 1: Abstract structure for incremental algorithms

value of the interaction of user u with item i is given by $\hat{r}_{ui} = \mathbf{a}_u \mathbf{b}_i^{\mathrm{T}}$. Incremental approaches already exist, as the one presented by Vinagre et al. (2014a) which uses stochastic gradient descent (SGD) as a search function it also devises a mechanism to give greater importance to more recent events, it is called Recency Adjust ISGD (RAISGD) and is the chosen algorithm in this work to represent the Matrix Factorization class of algorithms as a proved baseline for non-context aware techniques, its objective function is:

$$\min_{\mathbf{A},\mathbf{B}} \sum_{(u,i)\in D} (1 - \hat{r}_{ui})^2 + \lambda \left(|\mathbf{a}_u|^2 + |\mathbf{b}_i|^2 \right), \tag{1}$$

Where $1 - \hat{r}_{ui}$ is the error term and $\lambda \geq 0$ a regularization parameter. The incremental update steps for SGD are:

$$\begin{aligned} \mathbf{a}_u &\leftarrow \mathbf{a}_u + \eta \left[(1 - \hat{r}_{ui})\, \mathbf{b}_i - \lambda \mathbf{a}_u \right] \\ \mathbf{b}_i &\leftarrow \mathbf{b}_i + \eta \left[(1 - \hat{r}_{ui})\, \mathbf{a}_u - \lambda \mathbf{b}_i \right] \end{aligned} \tag{2}$$

As such, and considering the k and D in Algorithm 1 we have a linear complexity for this model of $O(kN)$ where $N = |D|$, the length of the dataset.

2.2. Matrix Co-Factorization

Following the above idea, Co-factorization techniques use contextual information to improve the recommendations. The main idea is that of considering four matrices $\mathbf{A}$ (user matrix), $\mathbf{B}$ (item matrix), $\mathbf{X}$ (users' attribute matrix) and $\mathbf{Y}$ (items' attribute matrix) that cover the common k-dimensional latent space, having $\mathbf{S} \approx \mathbf{A}\mathbf{X}^{\mathrm{T}}$ and $\mathbf{T} \approx \mathbf{B}\mathbf{Y}^{\mathrm{T}}$. Also, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{X}$ and $\mathbf{Y}$ that minimize the L^2 loss function:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{X},\mathbf{Y}} \left|\left|\mathbf{R} - \mathbf{A}\mathbf{B}^{\mathrm{T}}\right|\right|_F^2 + \left|\left|\mathbf{S} - \mathbf{A}\mathbf{X}^{\mathrm{T}}\right|\right|_F^2 + \left|\left|\mathbf{T} - \mathbf{B}\mathbf{Y}^{\mathrm{T}}\right|\right|_F^2. \tag{3}$$

An incremental technique based on RAISGD has been developed by Anyosa et al. (2018). This concrete implementation is suited for implicit feedback situations, so it is adequate for situations where there is a lack of explicit negative feedback and we can assume all the events are positive ones. This technique is called CORAISGD and was selected for this works' experiments as the first representative of Co-factorization techniques. In this paper we abbreviate the technique's name to CORA.

Focusing only on the learning logic behind CORA and considering that $\mathbf{X} \in \mathbb{R}^{p\times k}$ and $\mathbf{Y} \in \mathbb{R}^{q\times k}$. Let G be the set that contains all attributes' values of $\mathbf{X}$ and let H be the set that contains all attributes' values of $\mathbf{Y}$. Then, $\#(G) = p$ and $\#(H) = q$. For a given (u, i) tuple:

- Let G_u be the set of attributes' values that user u has.
- Let H_i be the set of attributes' values that item i has.
- Determine $\mathbf{X}_u \in \mathbb{R}^{\#(G_u)\times k}$, a submatrix of $\mathbf{X}$, where each row of $\mathbf{X}_u$ is the corresponding $\mathbf{x}_g = [x_{g1}, x_{g2}, \ldots, x_{gk}]$ row of $\mathbf{X}$, where $g \in G_u$
- Determine $\mathbf{Y}_i \in \mathbb{R}^{\#(H_i)\times k}$, a submatrix of $\mathbf{Y}$, where each row of $\mathbf{Y}_i$ is the corresponding $\mathbf{y}_h = [y_{h1}, y_{h2}, \ldots, y_{hk}]$ row of $\mathbf{Y}$, where $h \in H_i$

As done in MF case, for incremental implementations we optimize over the known values (u, i) of dataset D. The regularized and weighted loss function is given by:

$$\begin{aligned} \min_{\mathbf{A},\mathbf{B},\mathbf{X},\mathbf{Y}} \sum_{(u,i)\in D} \Bigg[& \omega_1 \left(1 - \hat{r}_{ui}\right)^2 + \frac{\omega_2}{\#(G_u)} \left|\mathbf{1} - \mathbf{a}_u \mathbf{X}_u^{\mathrm{T}}\right|^2 + \frac{\omega_3}{\#(H_i)} \left|\mathbf{1} - \mathbf{b}_i \mathbf{Y}_i^{\mathrm{T}}\right|^2 \\ & + \lambda \left(|\mathbf{a}_u|^2 + |\mathbf{b}_i|^2 + \frac{1}{\#(G_u)} ||\mathbf{X}_u||_F^2 + \frac{1}{\#(H_i)} ||\mathbf{Y}_i||_F^2 \right) \Bigg], \end{aligned} \tag{4}$$

where $\omega_1, \omega_2, \omega_3$ are weight parameters, λ is the regularization parameter and the other values are the same as considered before.

Incremental Matrix Co-Factorization optimizes using SGD, for each (u, i) we follow the gradient of (4), using the update expressions given below:

$$
\begin{aligned}
\mathbf{a}_u &\leftarrow \mathbf{a}_u + \eta \left[\omega_1 \left(1 - \hat{r}_{ui}\right) \mathbf{b}_i + \frac{\omega_2}{\#(G_u)} \left(\mathbf{1} - \mathbf{a}_u \mathbf{X}_u^{\mathrm{T}}\right) \mathbf{X}_u - \lambda \mathbf{a}_u \right] \\
\mathbf{b}_i &\leftarrow \mathbf{b}_i + \eta \left[\omega_1 \left(1 - \hat{r}_{ui}\right) \mathbf{a}_u + \frac{\omega_3}{\#(H_i)} \left(\mathbf{1} - \mathbf{b}_i \mathbf{Y}_i^{\mathrm{T}}\right) \mathbf{Y}_i - \lambda \mathbf{b}_i \right] \\
\mathbf{X}_u &\leftarrow \mathbf{X}_u + \eta \left[\frac{\omega_2}{\#(G_u)} \left(\mathbf{1} - \mathbf{a}_u \mathbf{X}_u^{\mathrm{T}}\right)^{\mathrm{T}} \mathbf{a}_u - \frac{\lambda}{\#(G_u)} \mathbf{X}_u \right] \\
\mathbf{Y}_i &\leftarrow \mathbf{Y}_i + \eta \left[\frac{\omega_3}{\#(H_i)} \left(\mathbf{1} - \mathbf{b}_i \mathbf{Y}_i^{\mathrm{T}}\right)^{\mathrm{T}} \mathbf{b}_i - \frac{\lambda}{\#(H_i)} \mathbf{Y}_i \right].
\end{aligned}
\tag{5}
$$

Just as Machine Factorization, so does this Matrix Co-Factorization technique have a linear complexity of $O(kN)$ considering, however, the existence of more update steps which reflect in a scale factor that does not increase the complexity but may affect execution time.

2.3. Factorization Machines

Factorization Machines (Rendle, 2010) is a technique that is not recommender system-specific but that can be used effectively for this purpose. These are inspired by classical Support Vector Machines and work with feature vectors which, in the context of the datasets used in Section 4, consist of $\{user, item, context\}$. An incremental version has been proposed by Kitazawa (2016), focusing on recommendations. Although the reported approach tackles cold start – a typical problem in recommender systems – by performing batch training on 30% of the data, in this work we do not make the same decision, so as to make the comparison between algorithms more fair.

Considering the learning process of incremental factorization machines we have the traditional learning model as:

$$
\hat{y}(\mathbf{x}) := \underbrace{w_0}_{\textbf{global bias}} + \underbrace{\mathbf{w}^{\mathrm{T}}\mathbf{x}}_{\textbf{linear}} + \sum_{i=1}^{d} \sum_{j=i}^{d} \underbrace{\mathbf{v}_i^{\mathrm{T}} \mathbf{v}_j}_{\textbf{interaction}} x_i x_j
$$

Figure 2: Factorization Machines predictor

Where x is the input vector containing $\{user, item, context1, context2, ...\}$, d is the number of context attributes used (1 for the case of our experiments), $\mathbf{V} \in \mathbb{R}^{d \times k}$ is a rank-k matrix. Kitazawa (2016) has proposed an incremental learning process for this estimator.

Non-incremental Factorization Machines have an original complexity of $O(kN^2)$ that can, however, be reformulated to avoid exhaustive computations and reduced to $O(kN)$ (Rendle, 2010). This is precisely the linear complexity achieved by the aforementioned proposal for Incremental Factorization Machines (Kitazawa, 2016).

2.4. CANDECOMP/PARAFAC (CP) Tensor Factorization

In an effort to create an incremental tensor factorization algorithm inspired by non-incremental CP tensor decomposition – see (Hitchcock, 1927) –, the current section describes a new incremental approach to tensor factorization. However, the preliminary results with the datasets used in this study were unsatisfying and are therefore not presented in Section 4.

Nevertheless the overall approach consists of incrementally factorizing a tensor $R \in \mathbb{R}^{M\times N\times C}$ (see Figure 3) into three matrices $\mathbf{A} \in \mathbb{R}^{M\times K}$, $\mathbf{B} \in \mathbb{R}^{N\times K}$ and $\mathbf{C} \in \mathbb{R}^{C\times K}$ that cover a common K-dimensional latent space, such that: $R \approx [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$. Matrix $\mathbf{A}$ spans the user space, $\mathbf{B}$ spans the item space and $\mathbf{C}$ the context space.

For the presented incremental implementation, we have the following regularized function to minimize:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \sum_{(u,i,f)\in D} (1-\hat{r}_{uif})^2 + \lambda \left(|\mathbf{a}_u|^2 + |\mathbf{b}_i|^2 + |\mathbf{c}_f|^2 \right), \tag{6}$$

Considering this, an SGD-based optimization can be executed with the obtained gradients:

$$\begin{aligned} \mathbf{a}_u &\leftarrow \mathbf{a}_u + \eta \left[(1-\hat{r}_{uif})\, \mathbf{b}_i * \mathbf{c}_f - \lambda \mathbf{a}_u \right] \\ \mathbf{b}_i &\leftarrow \mathbf{b}_i + \eta \left[(1-\hat{r}_{uif})\, \mathbf{a}_u * \mathbf{c}_f - \lambda \mathbf{b}_i \right] \\ \mathbf{c}_f &\leftarrow \mathbf{c}_f + \eta \left[(1-\hat{r}_{uif})\, \mathbf{a}_u * \mathbf{b}_i - \lambda \mathbf{c}_f \right] \end{aligned} \tag{7}$$

A linear time complexity is also expected from this algorithm, which is in agreement with the time complexity for the non-incremental techniques on which it is based (Rendle and Schmidt-Thieme, 2010).

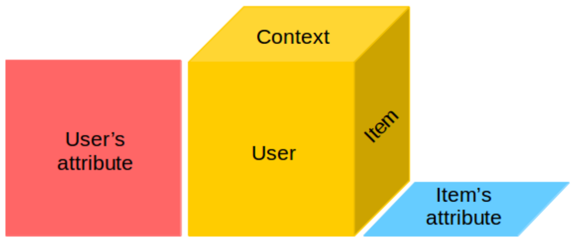

Figure 3: Tensor visualization of user-item-context tuples for factorization models

2.5. Pairwise Interaction Tensor Factorization

A well established approach for RS using tensors is the Pairwise Interaction Tensor Factorization (PITF), which is a special case of the Tucker Decomposition (R Tucker, 1966) with linear runtime for learning and prediction (Rendle and Schmidt-Thieme, 2010). Furthermore, the originally proposed version's optimization criterion is based on Bayesian Personalized Ranking (BPR). The original version includes an outer for-loop for each of the $max_iterations$ and inside it there is an inner-loop that iterates every data point and performs the necessary updates to the model. Algorithm 2 shows the exact opposite: first each data point is iterated and then the model is updated once for every value in $max_iterations$. The consequence is that this is now an incremental version and one can indeed perform both incremental evaluations (see Section 3 for further detail) as well as predictions at any point in the dataset, which was impossible up to now, given that the model had to go over the entire dataset multiple times. This is, then, an Incremental PITF (IPITF) and corresponds to the version proposed and analyzed in Section 4.

```
**Input** : $k$ ($max_iterations$), $\lambda$, $\eta$
for $(u,i,f) \in D$ do
    Perform optional prediction + evaluation
    for $t \leftarrow 1$ to $k$ do
        $\hat{u}_{u,f} \leftarrow \hat{u}_{u,f} + \eta(\delta * (\hat{t}^U_{t_{A,f}} - \hat{t}^U_{t_{B,f}}) - \lambda * u_{u,f})))$
        $\hat{i}_{i,f} \leftarrow \hat{i}_{i,f} + \eta(\delta * (\hat{t}^I_{t_{A,f}} - \hat{t}^I_{t_{B,f}}) - \lambda * i_{i,f})))$
        $\hat{t}^U_{A,f} \leftarrow \hat{t}^U_{A,f} + \eta(\delta\hat{u}_{u,f} - \lambda\hat{t}^U_{A,f})$
        $\hat{t}^U_{B,f} \leftarrow \hat{t}^U_{B,f} + \eta(-\delta\hat{u}_{u,f} - \lambda\hat{t}^B_{A,f})$
        $\hat{t}^I_{A,f} \leftarrow \hat{t}^I_{A,f} + \eta(\delta\hat{i}_{i,f} - \lambda\hat{t}^I_{A,f})$
        $\hat{t}^I_{B,f} \leftarrow \hat{t}^I_{B,f} + \eta(-\delta\hat{i}_{i,f} - \lambda\hat{t}^I_{B,f})$
    end
end
```

Algorithm 2: Incremental Pairwise Interaction Tensor Factorization

With each $\hat{t}$ and $\hat{u}$ representing the gradients for the PITF update, η is the learning rate and λ the regularization parameters, δ is the prediction error. Original PITF has a complexity that is linear according to the number of factorization dimensions used the incremental version does not increase the complexity only the order of operations (Rendle and Schmidt-Thieme, 2010).

3. Incremental Evaluation of Recommendations

3.1. Prequential Evaluation

Considering the aforementioned algorithms perform incremental learning over data streams, it is necessary to use a suitable evaluation process (Gama et al., 2009). In our experiments, we use the prequential evaluation protocol studied by Vinagre et al. (2014b).

As each (u, i) (or (u, i, c)) event arrives in the stream, we first use it to test the model, and then we use it to update the model. We perform the following steps for each new event:

1. If u is a known user (and the context c is also known, where applicable), use the current model to recommend a list of items to u, otherwise go to step 3;
2. Score the recommendation list given the observed item i;
3. Update the model with (u, i) (or (u, i, c));
4. Proceed to the next observation.

3.2. Metrics

We report accuracy using the HitRate@N measure at cut-offs of N$\in \{1, 5, 10, 20\}$. The HitRate@N$= 1$ if item i is in the first N recommended items, and equals 0 otherwise. To obtain the top-N items, we previously score all the available items using $\hat{r}_{ui}$, order them, and select those with the highest values.

Additionally, we present selected plots of the moving average of HitRate@20 that is computed considering an averaged mean over batches of size $n = 5000$, that was chosen to foster illustrative information visualization. We also report the average update time per tuple processed by each algorithm to allow for a comparison between time complexity and execution time.

3.3. Statistical Tests

Our incremental implementations provide us with a sequence of the hit rate values that represent the learning process of the models. We can use sliding windows to report the mean of the hit rate continuously. Hence, in each window we are going to have a sequence of hit rate $\in \{0, 1\}$ of size n that is compatible with the case of the 0-1 loss function described by Gama et al. (2009). Consequently, we use the McNemar test in each window, to compare the incremental learning performance of the different models.

The test defines that given two algorithms A and B, we count the number of times n_{10} where the prediction of A is correct and the prediction of B is wrong and the number of times n_{01} where we have the opposite situation. The statistic of the test is given by:

$$M = \frac{(n_{10} - n_{01})^2}{n_{10} + n_{01}}, \tag{8}$$

where $M \sim \chi^2_1$ and the critical value for a significance level of $\alpha = 0.01$ is $M = 6.635$. As we can see, the computations are simple and can be easily implemented over a finished stream approach.

4. Experiments

4.1. Datasets

Five datasets were selected for experimental analysis, three from the music field, one from apparel sales and another from general purpose sales, as follows:

- Last.fm: obtained from a music service website with the same name (Bertin-Mahieux et al., 2011) (LFM)
- Palco Principal: obtained from a social network of the same name and collected by Vinagre et al. (2014a) (PLC)
- #nowplaying: obtained from the Twitter users' posts of current listened songs and gathered by Zangerle et al. (2014) (NP)
- RentTheRunway: obtained from the reviews in an online platform that allows women to rent clothes for various occasions (Misra et al., 2018) (RTR)
- Epinions: ratings for general consumer reviews (Zhao et al., 2014) (EPI)

Table 2 describes the number of events, users and items in each dataset as well as the number of unique attributes in the context dimension, and the dataset sparsity.

Dataset	Events	Users	Items	Attributes	Sparsity (%)
LFM	403798	280	196734	21566	99.999966
PLC	508705	20875	25262	5163	99.999981
NP	444086	4131	175014	36519	99.999998
RTR	177928	101233	5801	68	99.999554
EPI	119158	73794	33973	2150	99.999998

Table 2: Dataset Description

4.2. Experimental Setup

We tuned the hyper-parameters manually, using the first 25% of the instances of each dataset. For each hyper-parameter, several values were selected and tested while fixing the remaining parameters. After all the results were reported, we selected each parameter individually according to the value that performed better, using the HitRate@N values as mentioned above, for that purpose. This assumes a certain level of independence between parameters but proved a valid approach as the results were in agreement with previous reported work.

For complementary pairs of algorithms, namely RAISGD and CORA, as well as contextless iFM and iFM, the same hyperparameters were used, namely the ones calculated for the context-aware version. This was so that an estimate of the impact of the extra dimension could be measured instead of tuning for both settings, which would result in lack of consistency for comparing these approaches among themselves and against the remaining ones.

Afterwards, we used the chosen hyper-parameters to evaluate the algorithmic performance in the 100% of the datasets and report the results. The experiments were ran on a 32-core cluster of Intel Core Processor (Haswell, no TSX) and in a Python 3.6 environment.

4.3. Results

Table 3 summarizes the overall performance of the multiple algorithms across the selected datasets. From this table we can make multiple observations:

- IPITF has the best predictive power
- iFm and CORA present similar results, which is in accordance with their similar nature. This observation is even more clear in Figures 4 and 5.
- Likewise, contextless iFM and RAIGSD behave similarly, also in accordance with their context-aware versions and their shared nature

Additional experiments were executed for the context-aware algorithms were ran on noisy datasets - original datasets where the context dimension was randomly generated without changing the cardinality of that dimension in each of the original datasets. In those experiments, iFM and CORA produced poor results (HitRate@20 $\approx$ 0) reflecting these algorithms problems in handling noise. In contrast, IPITF proved to be somewhat robust to noise. The difference in the average values for the HitRate@20 is as described in

	RAISGD				**CORAISGD**			
HitRate@	**1**	**5**	**10**	**20**	**1**	**5**	**10**	**20**
lfm	0.00000	0.04051	0.06027	0.06889	0.00025	0.04051	0.05885	0.07411
plc	0.00006	0.29242	0.33716	0.36368	0.05154	0.31932	0.43586	0.51479
np	0.00028	0.00133	0.00292	0.00621	0.00003	0.00751	0.11808	0.15790
epinions	0.00004	0.00036	0.00060	0.00146	0.00004	0.00040	0.00069	0.00122
renttr	0.00014	0.00103	0.00190	0.00366	0.00004	0.00055	0.00114	0.00212
	iFM No Context				**iFM**			
HitRate@	**1**	**5**	**10**	**20**	**1**	**5**	**10**	**20**
lfm	0.00004	0.00041	0.00347	0.01949	0.00007	0.00224	0.01658	0.05440
plc	0.00034	0.01714	0.11939	0.34074	0.00242	0.05613	0.24665	0.52018
np	0.00006	0.00046	0.00362	0.02483	0.00005	0.00171	0.01950	0.08683
epinions	0.00016	0.00047	0.00129	0.00297	0.00036	0.00062	0.00091	0.00144
renttr	0.00003	0.00025	0.00098	0.00254	0.00003	0.00033	0.00080	0.00153
	Not applicable				**IPITF**			
HitRate@					**1**	**5**	**10**	**20**
lfm					0.04105	0.13207	0.20262	0.30006
plc					0.94084	0.96192	0.96863	0.97452
np					0.37141	0.49350	0.54430	0.59707
epinions					0.12199	0.24846	0.32191	0.40196
renttr					0.02888	0.09345	0.13842	0.20245

Table 3: Average hit rates for all the datasets, all algorithms and hit rate cutoffs of 1,5,10 and 20. The main left column is for algorithms that are not context aware and right column for those that are.

Table 4, it is clear that the results in the original dataset are better but that the addition of noise only affects the predictive power by an offset that can be attributed to the information contained in the context dimension, that is lost in the noisy datasets.

Dataset	**original - noisy**
lfm	0.053943
plc	0.057924
np	0.134431

Table 4: Difference between HitRate@20 average values for IPITF without and with noise in the context dimension (for the lfm, plc and np datasets)

Figures 4 and 5 provide a more detailed comparison specific to the lfm and plc datasets, respectively. Plots from the remaining datasets are not included as there is little to gain in terms of novel information.

Figures 6 and 7 display the results of a McNemar Test over a sliding window, with a significance of 1%, for CORA and iFM. Although these algorithms are somewhat similar in

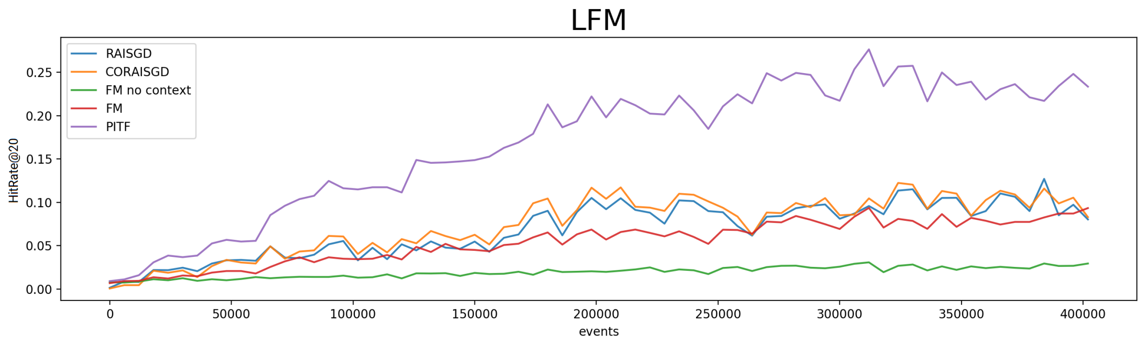

Figure 4: Evolution of HitRate@20 as events are seen, for the lfm dataset

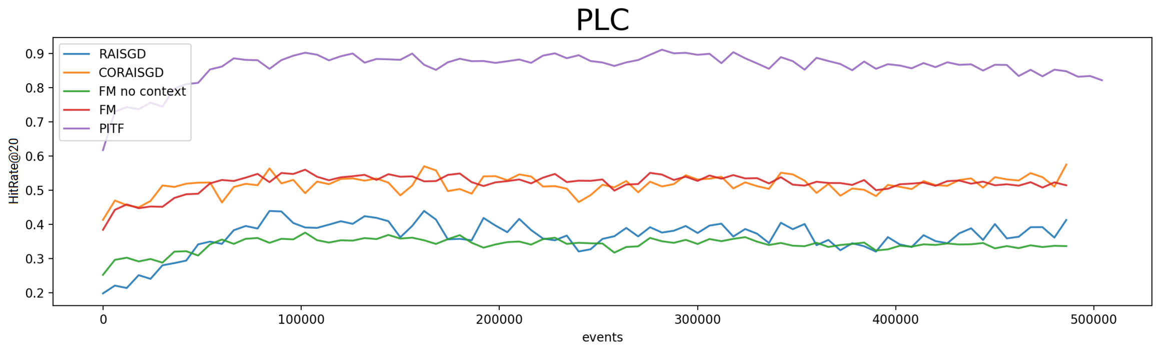

Figure 5: Evolution of HitRate@20 as events are seen, for the plc dataset

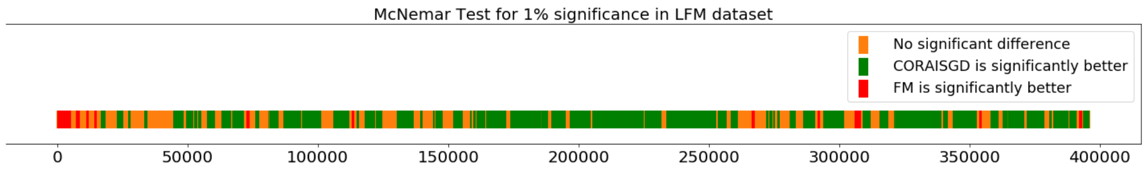

Figure 6: McNemar tests comparison for CORA and iFM for the lfm dataset

terms of predictive power, CORA still reveals to be either as good or significantly better than iFMs, regardless of the dataset.

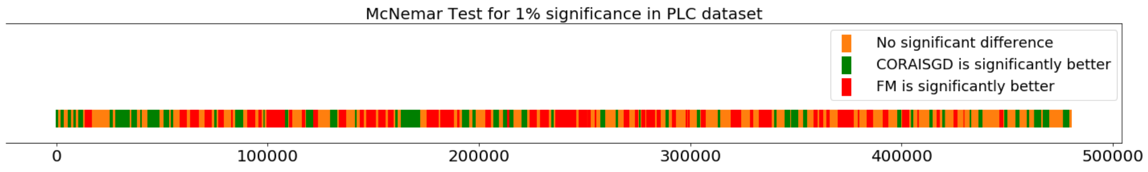

Figure 7: McNemar tests comparison for CORA and iFM for the plc dataset

The execution time of the algorithms - amount of time needed to updated and predict for one event - is consistent across all the datasets. The average execution time values are presented in Table 5. Their values, in seconds and in decreasing order are as follows: 0.88s for iFM, 0.25s for IPITF, 0.022 for CORA and 0.021 for RAISGD. With the iFM implementation being significantly slower than that of CORA and RAISGD - 40 times slower processing each individual event. IPITF is also slower than CORA and RAISGD by a factor of approximately 11.

Algorithm	**Execution Time (s)**
iFM	0.88
IPITF	0.25
CORA	0.022
RAISGD	0.021

Table 5: Average Execution Time across algorithms

5. Conclusions

With this survey, four different algorithms for incremental recommender systems have successfully been compared, two of which work for context-aware situations (CORA, IPITF), one for contextless scenarios (RAISGD) and one for both (iFM). Efforts for developing a new tensor-inspired incremental approach were also presented (incten). IPITF is also a part of the proposed novelties, as an incremental version of the existent PITF approach. IPITF proved to have surpassed all other algorithms in terms of predictive power and resilience to noise.

In terms of Matrix Factorization-inspired approaches, CORA proved to be either as good or better than iFM and both surpassed the respective contextless approaches. Likewise it is reported that both these approaches are susceptible to noise in the context dimension.

In terms of time complexity, all algorithms fall into the linear class, a typical requirement for incremental learning algorithms. However, the execution times differ significantly with iFM proving to be slowest by a factor of 80 to RAISGD and CORA and by a factor of 3.5 to IPITF.

Acknowledgments

This work is financed by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia within project: UID/EEA/50014/2019.

References

Susan C. Anyosa, João Vinagre, and Alípio M. Jorge. Incremental matrix co-factorization for recommender systems with implicit feedback. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 1413–1418, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-5640-4. doi: 10.1145/3184558.3191585. URL https://doi.org/10.1145/3184558.3191585.

Robert M Bell and Yehuda Koren. Lessons from the netflix prize challenge. *Acm Sigkdd Explorations Newsletter*, 9(2):75–79, 2007.

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 329–338. ACM, 2009.

Frank L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

Takuya Kitazawa. Incremental factorization machines for persistently cold-starting online item recommendation, 2016.

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.

Fangfang Li, Guandong Xu, and Longbing Cao. Coupled item-based matrix factorization. In *International Conference on Web Information Systems Engineering*, pages 1–14. Springer, 2014.

Rishabh Misra, Mengting Wan, and Julian McAuley. Decomposing fit semantics for product size recommendation in metric spaces. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 422–426, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5901-6. doi: 10.1145/3240323.3240398. URL http://doi.acm.org/10.1145/3240323.3240398.

Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 02 1966. doi: 10.1007/BF02289464.

Steffen Rendle. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 995–1000, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4256-0. doi: 10.1109/ICDM.2010.127. URL http://dx.doi.org/10.1109/ICDM.2010.127.

Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 81–90, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-889-6. doi: 10.1145/1718487.1718498. URL http://doi.acm.org/10.1145/1718487.1718498.

Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 635–644. ACM, 2011.

M Venkatesan and K Thangadurai. History and overview of the recommender systems. *Collaborative Filtering Using Data Mining and Analysis*, page 74, 2016.

João Vinagre, Alípio Mário Jorge, and João Gama. Fast incremental matrix factorization for recommendation with positive-only feedback. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 459–470. Springer, 2014a.

João Vinagre, Alípio Mário Jorge, and João Gama. Evaluation of recommender systems in streaming environments. In *Proceedings of the Workshop on Recommender Systems Evaluation: Dimensions and Design in conjunction with the 8th ACM Conference on Recommender Systems (RecSys 2014), Foster City, CA, USA, October 10, 2014.*, 2014b.

Wolfgang Woerndl and Johann Schlichter. Introducing context into recommender systems. In *Proceedings of AAAI workshop on recommender systems in E-commerce*, pages 138–140, 2007.

Eva Zangerle, Martin Pichl, Wolfgang Gassler, and Günther Specht. nowplaying music dataset: Extracting listening behavior from twitter. In *Proceedings of the 1st ACM International Workshop on Internet-Scale Multimedia Management*, ISMM '14, pages 21–26, New York, NY, USA, June 2014. ACM.

Tong Zhao, Julian McAuley, and Irwin King. Leveraging social connections to improve personalized ranking for collaborative filtering. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 261–270, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2598-1. doi: 10.1145/2661829.2661998. URL http://doi.acm.org/10.1145/2661829.2661998.