Incremental Matrix Co-factorization for Recommender Systems with Implicit Feedback

Susan C. Anyosa
LIAAD - INESC TEC
Porto, Portugal
scanyosa@inesctec.pt

João Vinagre
LIAAD - INESC TEC
FCUP, University of Porto
Porto, Portugal
jnsilva@inesctec.pt

Alípio M. Jorge
LIAAD - INESC TEC
FCUP, University of Porto
Porto, Portugal
amjorge@fc.up.pt

ABSTRACT

Recommender systems try to predict which items a user will prefer. Traditional models for recommendation only take into account the user-item interaction, usually expressed by explicit ratings. However, in these days, web services continuously generate auxiliary data from users and items that can be incorporated into the recommendation model to improve recommendations. In this work, we propose an incremental Matrix Co-factorization model with implicit user feedback, considering a real-world data-stream scenario. This model can be seen as an extension of the conventional Matrix Factorization that includes additional dimensions to be decomposed in the common latent factor space. We test our proposal against a baseline algorithm that relies exclusively on interaction data, using prequential evaluation. Our experimental results show a significant improvement in the accuracy of recommendations, after incorporating an additional dimension in three music domain datasets.

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Computing methodologies → Factorization methods**; • **Theory of computation → Online algorithms**;

KEYWORDS

Recommender Systems; Matrix Co-Factorization; Implicit feedback; Incremental Learning; Data Streams

ACM Reference Format:
Susan C. Anyosa, João Vinagre, and Alípio M. Jorge. 2018. Incremental Matrix Co-factorization for Recommender Systems with Implicit Feedback. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3184558.3191585

1 INTRODUCTION

Recommender systems (RS) are well-known for being used on e-commerce websites like Amazon[1], on video recommendations like Youtube[2] and streamming programs such Netflix[3]. They attempt to predict customers responses in order to guide them through the wide variety of offered items.

RS can be built using two fundamentally different strategies: content-based filtering and collaborative filtering (CF). The first one recommends items considering their intrinsic or descriptive information. The second one exploits past interactions between users and items to select the items that best match each user's preferences.

Considering the CF approach, there are two main model classes: neighborhood-based and matrix factorization. The first class relies on relationships between either items (item-based) or users (user-based) using a similarity measure. K-nearest neighbors (kNN) algorithm is one of the most popular in CF and are usually used as a benchmark in comparisons. The second class describes both items and users in a common latent factor space using matrix decomposition methods. Currently, most state-of-the art algorithms adopt MF models.

One popular strategy to make MF models more accurate is to use side information, typically available in real-world applications. Recent proposals adapt traditional MF to incorporate the information of additional dimensions, such as users or items features [4, 16]. Another approach is Matrix Co-factorization (MCF) [6, 10, 21], in which the additional dimensions are also factorized onto the same shared latent feature space of users and items.

Another important aspect of recommendation problems is the type of feedback, and the rate at which it is generated. In RS problems, the user-item interaction can be expressed by explicit ratings given by the users, known as explicit feedback. However, in real-world scenarios, explicit feedback is not always available. In this situation user preferences are inferred from implicit feedback, such as purchase history, browsing history, search patterns, clicks on items, etc. Moreover, this kind of feedback is continuously being generated, at potentially fast rates. In the era of big data, it is important to take into account that RS users generate user feedback continuously, as they interact with items. New users, items, and features are added constantly. In this setting, it is necessary to learn from data without recomputing the models again every time new data is available. Instead, we can achieve this considering an incremental learning approach and assess the incremental modeling with suitable stream-based methodologies [7].

To address the situations described before, in this paper, we propose the first incremental MCF formulation and algorithm for recommendation in presence of implicit feedback. Also, we present

[1]https://www.amazon.com

[2]https://www.youtube.com
[3]https://www.netflix.com

an empirical comparison of the online behavior of MCF with additional features against incremental MF without additional features. Using proper stream-based evaluation, when compared with MF, our incremental MCF implementation shows superior accuracy.

This paper is structured as follows. In the following section we review the related work. In Sec. 3 we review useful preliminaries on MF model and notations. In Sec. 4 we present our incremental MCF formulation and algorithm. The experiments and results are reported in Sec. 5. Finally, we conclude in Sec. 6.

2 RELATED WORK

In this section, we categorize the related work into the following topics:

Additional dimensions. RS models that incorporate content (features) or context information to improve predictions are vast in the literature. In the group of matrix decomposition, Singh and Gordon [21] proposed the MCF model (also known as Collective Matrix Factorization) using a solid theoretical support. Later, an application of the former work, was made by Fang and Si [6]. Also, an extension applied to the analysis of social network data was made by Hong et al. [10]. Using another approach, Li et al. [16] showed that feature data can also be incorporated inside a MF model, previously using an attribute-based similarity measure between items. Additionally, using a linear model approach, Chen et al. [4] included feature data in a MF model. Domingues et al. [5] treated new dimensions as virtual items which are processed as regular items by algorithms such as association rules or neighborhood based approaches. In the group of tensor models, a natural way to include additional dimensions is considering them as modes [9, 13, 22].

Incremental learning. CF systems that provide a real-time response to the big data challenges have been developed recently [3, 12]; other proposals consider a probabilistic model, such as [27]. Additionally, MF incremental algorithms have been developed by Gemulla et al. [8], Pálovics et al. [17], Sarwar et al. [20], Takács et al. [23], Vinagre et al. [25], Zheng and Xie [29]. Tensor dynamic implementations were also provided by Kasai [14], Song et al. [22], Zhou et al. [30].

Implicit feedback. Some authors studied RS algorithms that address implicit user feedback, also known as one-class collaborative filtering [11, 18]. Pilászy et al. [19] studied an improved MF model using Alternating Least Squares optimization, and as an extension, Hidasi and Tikk [9] developed the tensor version. Also, Vinagre et al. [25, 26] worked with one-class feedback.

In this stage of our work we are interested in evaluating the recommendation improvement obtained after incorporating additional dimensions, for implicit user feedback in a stream-based learning setup. For this reason, we focus on comparisons with the RAISGD algorithm [26].

3 PRELIMINARIES

In this section we briefly describe the MF procedure since a similar approach is followed for MCF. The notation along this work is presented in Table 1.

MF and MCF models consider an entity-relation schema. For example, in a music domain problem, the entities may be users, songs and artists. These entities have relations between them that can be represented with matrices. In simple MF, only the information of the relation user-item is considered. This can be represented with the matrix $\mathbf{R}$. $\mathbf{R} \in \mathbb{R}^{m \times n}$ is also known as *feedback matrix*, with m users and n items. Each value of $\mathbf{R}$ is defined by:

Table 1: Table of symbols.

Notations	Definitions
$\mathbf{A}$, $\mathbf{a}$, a	Matrix, vector, scalar
$\lvert\mathbf{a}\rvert$	Vector norm (ℓ^2-norm) of $\mathbf{a}$
$\lvert\lvert\mathbf{A}\rvert\rvert_F$	Frobenius norm of $\mathbf{A}$
$\#(S)$	Cardinality of set S

$$r_{ui} = \begin{cases} 1 & \text{if user } u \text{ interacts with item } i, \\ 0 & \text{otherwise.} \end{cases}$$

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. The predicted value of the interaction of user u with item i is given by the following product:

$$\hat{r}_{ui} = \mathbf{a}_u \mathbf{b}_i^{\mathrm{T}} = [a_{u1}, a_{u2}, \ldots, a_{uk}]\,[b_{i1}, b_{i2}, \ldots, b_{ik}]^{\mathrm{T}}\,. \tag{1}$$

In MF algorithms, we want to find the optimal $\mathbf{A}$ and $\mathbf{B}$ that minimize the L^2-regularized squared error, given by:

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

where D is the set containing each (u, i) known pair – i.e. where $r_{ui} = 1$ –, the error term is $1 - \hat{r}_{ui}$ and $\lambda \geq 0$ is the regularization parameter. The formula on (2) was initially popularized by Simon Funk[4], and later clearly detailed by Takács et al. [23]. The most popular learning methods used to solve this optimization problem are Alternating Least Squares (ALS) [1] and Stochastic Gradient Descent (SGD). Koren et al. [15] gives an introductory description of both methods.

In this paper we use SGD, which starts initializing $\mathbf{A}$ and $\mathbf{B}$ with random values from $N(\mu, \sigma)$, where $\mu = 0$ and small σ. In the batch mode, SGD performs several iterations over all the available tuples (u,i), until a stopping criterion is met. At each iteration, SGD updates $\mathbf{a}_u$ and $\mathbf{b}_i$, correcting them in the opposite direction of the gradient of (2) by a factor of the learn rate $\eta < 1$. For each (u, i) known pair, the following updates are performed:

$$\begin{aligned} \mathbf{a}_u &\leftarrow \mathbf{a}_u + \eta\left[(1-\hat{r}_{ui})\,\mathbf{b}_i - \lambda\mathbf{a}_u\right] \\ \mathbf{b}_i &\leftarrow \mathbf{b}_i + \eta\left[(1-\hat{r}_{ui})\,\mathbf{a}_u - \lambda\mathbf{b}_i\right] \end{aligned} \tag{3}$$

An incremental version of SGD (ISGD) is proposed by Vinagre et al. [25].

4 INCREMENTAL MATRIX CO-FACTORIZATION

Following the example of music domain in Sec. 3, in MCF, we use extra information, considering also the song-artist relation. The idea is to jointly factorize users, items, user features and item

[4]http://sifter.org/ simon/journal/20061211.html

features. For each additional relation user-/item-feature, a new matrix is added alongside the feedback matrix. For simplicity, in the following definitions, we considered two added matrices; however, the extension to more than two can be easily done.

Let $\mathbf{S} \in \mathbb{R}^{m \times p}$ be a user-feature matrix, with m users and p possible values for the feature, where:

$$s_{ug} = \begin{cases} 1 & \text{if user } u \text{ has the feature value } g, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathbf{T} \in \mathbb{R}^{n \times q}$ be an item-feature matrix, with n items and q possible values for the feature, where:

$$t_{ih} = \begin{cases} 1 & \text{if item } i \text{ has the feature value } h, \\ 0 & \text{otherwise.} \end{cases}$$

In Sec. 3, we saw that $\mathbf{R} \approx \mathbf{A}\mathbf{B}^{\mathrm{T}}$. In MCF, we decompose S into two low-rank matrices $\mathbf{A} \in \mathbb{R}^{m \times k}$ and $\mathbf{X} \in \mathbb{R}^{p \times k}$ as well. T is decomposed into two low-rank matrices $\mathbf{B} \in \mathbb{R}^{n \times k}$ and $\mathbf{Y} \in \mathbb{R}^{q \times k}$. Four matrices A (user matrix), B (item matrix), X (users' feature matrix) and Y (items' feature matrix) cover the common k-dimensional latent space. We have that $\mathbf{S} \approx \mathbf{A}\mathbf{X}^{\mathrm{T}}$ and $\mathbf{T} \approx \mathbf{B}\mathbf{Y}^{\mathrm{T}}$. We want to find the optimal A, B, X and Y that minimize:

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

where $\mathbf{1}$ is the unity vector, $\omega_1, \omega_2, \omega_3$ are weight parameters and the other values are the same as considered in Sec. 3. For a given user u and item i, G_u is the set of feature's values that user u has, and H_i is the set of feature's values that item i has. Also, let $\mathbf{X}_u$ and $\mathbf{Y}_i$ be sub-matrices of X and Y. $\mathbf{X}_u$ is obtained by selecting the $\mathbf{x}_g = \left[x_{g1}, x_{g2}, \ldots, x_{gk}\right]$ rows of X, where $g \in G_u$. $\mathbf{Y}_i$ is obtained by selecting the $\mathbf{y}_h = [y_{h1}, y_{h2}, \ldots, y_{hk}]$ rows of Y, where $h \in H_i$. Similarly to the SGD procedure in Sec. 3, for each (u, i) we follow the gradient of (4), using the update expressions given below:

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

To add more dimensions to the MCF model, we need to include the correspondent low-rank matrices in (4) and then obtain the update expressions as the ones presented in (5). Details of a batch implementation of MCF under ALS optimization can be found in [6]. Also, details about the convergence properties of the MCF model are given by [21].

Our incremental MCF algorithm using SGD does a single pass over the dataset performing the updates given by (5).

The implicit feedback only provides positives examples, which causes global convergence to the positive class and loss of accuracy [26]. To solve the lack of negative examples, we use recency-based negative feedback imputation proposed by Vinagre et al. [26]. A queue Q containing all the items seen in the stream is included. For each (u, i) in the stream, l negative examples $(u, j_1), \ldots, (u, j_l)$ are included for that user u, based on the recency of occurrence of the items. The $j_1, \ldots, j_l$ items are the l items that are in the tail of Q. Every time an item occurs, it is moved to the head of the queue. For more details on the method, please refer to [26].

The proposed algorithm (CORAISGD) is presented in Alg. 1, where the functions `initqueue`, `dequeue`, `enqueue` and `remove` perform queue initialization, tail removal, head insertion and index-based removal, respectively.

Algorithm 1: CORAISGD

```
Input  : $k$, $iterations$, $\lambda$, $\eta$, $l$, $\omega_1$, $\omega_2$, $\omega_3$
Output : A, B, X, Y
1  $Q \leftarrow$ initqueue()
2  for $(u,i) \in D$ do
3      if $u \notin$ rows(A) then
4          $\mathbf{a}_u \sim N(0, 0.1)$
5          for $g$ in $G_u$ do
6              if $g \notin$ rows(X) then
7                  $\mathbf{x}_g \sim N(0, 0.1)$
8      if $i \notin$ rows(B) then
9          $\mathbf{b}_i \sim N(0, 0.1)$
10         for $h$ in $H_i$ do
11             if $h \notin$ rows(Y) then
12                 $\mathbf{y}_h \sim N(0, 0.1)$
13     for $t \leftarrow 1$ to min($l$, size($Q$)) do
14         $j \leftarrow$ dequeue($Q$)
15         for $t \leftarrow 1$ to $iterations$ do
16             $\mathbf{a}_u \leftarrow \mathbf{a}_u + \eta[(0 - \hat{r}_{uj})\mathbf{b}_j - \lambda \mathbf{a}_u]$
17         enqueue($Q, j$)
18     for $t \leftarrow 1$ to $iterations$ do
19         Eq. (5)
20     if $i \in Q$ then
21         remove($Q, i$)
22     enqueue($Q, i$)
```

5 EXPERIMENTS AND EVALUATION

5.1 Prequential evaluation

Considering that CORAISGD performs incremental learning over data streams, it is necessary to use a suitable evaluation process [7]. In our experiments, we use the prequential evaluation protocol studied by Vinagre et al. [24].

As each (u, i) pair 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 (u, i) pair:

(1) If u is a known user, 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);
(4) Proceed to the next observation.

5.2 Datasets

All datasets in our experiments belong to the music domain and contain information about users, songs, artists and timestamp. We chose music domain datasets due to the great availability of resources. We sorted the records using the timestamp to simulate a real-world dynamic environment and feed them into our incremental algorithms. Besides the regular dimensions of users and items (songs), we took into account one additional dimension concerning items: artist. The datasets are:

- Last.fm was obtained from a music service website of the same name. The dataset was collected by Celma [2];
- Palco Principal was obtained from a social network of the same name and collected by Vinagre et al. [25];
- #nowplaying was obtained from the Twitter users' posts of current listened songs and gathered by Zangerle et al. [28].

We worked with a sample from the original datasets. Table 2 presents the number of values and the percentage of sparsity.

Table 2: Dataset description.

Dataset	Events	Users	Songs	Artists	Sparsity
Last.fm	403,798	280	196,734	21,566	99.27%
Palco Principal	508,705	20,875	25,262	5,163	99.90%
#nowplaying	444,086	4,131	175,014	36,519	99.94%

5.3 Experimental setup

As we mentioned in Sec. 2, in this work we compared CORAISGD with RAISGD [26]. Both are incremental algorithms that use *recency-based* imputation and deal with implicit feedback from users. We tuned the hyper-parameters manually using the first 25% of the instances of each dataset, selecting those that gave us higher accuracy considering the prequential evaluation protocol. 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 hp i7-16 GB desktop using Python 2.7.

5.4 Metrics

We report accuracy using the recall@N measure at cut-offs of N$\in$ $\{1, 5, 10, 20\}$. The recall= 1 if the item i is in the first N recommended items, and 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 plots of the moving average of recall@20 that is computed considering a sliding window of $n = 4000$, that was set for illustrative purposes.

We also report the update time per tuple processed by the algorithm.

5.5 Statistical test

Our incremental implementations provide us with a sequence of the recall values that represent the learning process of the models. As we mentioned before, we can use sliding windows to report the mean of the recall continuously. Hence, in each window we are going to have a sequence of recall $\in \{0, 1\}$ of size n that is compatible with the case of the 0-1 loss function described by Gama et al. [7]. Consequently, we use the McNemar test in each window, to compare the incremental learning performance of RAISG and CORAISGD.

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:

$$M = \frac{(n_{10} - n_{01})^2}{n_{10} + n_{01}}, \tag{6}$$

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 in a fully data-stream approach.

5.6 Results

The values of global average of recall and update time are presented in Table 3. We can see that CORAISGD has better accuracy than RAISGD in the three datasets. The update time is higher for CORAISGD for all the cases, but this was expected since it has more calculations involved. As we can see, analyzing only summary results give us a general idea of which algorithm is better, but not further details of the learning process as it occurs. However we can achieve it using stream-based prequential evaluation [24].

In Fig. 1 we show the moving average of recall@20, each point is the average recall of a sequence of $n = 4000$. This plot confirms the superiority of the proposed algorithm and consequently, the usefulness of taking into account additional feature data in the modeling. Furthermore, we can see how the recall evolves over time. For (b) Palco Principal and (c) #nowplaying the improvement is visible; however for (a) Last.fm it is not that easy. In such cases, we test if, in fact, a statistically significance difference between the performance of both algorithms exists.

Hence, we used the McNemar test in each sequence of $n = 4000$. In detail, we found that:

- For Last.fm: In 70.26% of the prequential process CORAISGD was better, in 2.06% RAISGD was better, and in 27.68% the difference between them was not statistically significant.
- For Palco Principal: In 100% of the prequential process CORAISGD was better.
- For #nowplaying: In 79.25% of the prequential process CORAISGD was better, in 8.63% RAISGD was better, and in 12.12% the difference between them was not statistically significant.

Fig. 2 presents illustrations of the outcome of the McNemar test, for each window, over time. We can see that CORAISGD was better in most of the windows.

Table 3: Results for RAISGD and CORAISGD.

Dataset	Last.fm		Palco Principal		#nowplaying	
Algorithm	RAISGD	CORAISGD	RAISGD	CORAISGD	RAISGD	CORAISGD
recall@1	<0.001	<0.001	0.004	0.010	<0.001	<0.001
recall@5	0.038	0.042	0.171	0.345	0.002	0.008
recall@10	0.057	0.060	0.225	0.463	0.003	0.012
recall@20	0.064	0.075	0.268	0.522	0.005	0.016
Update time (ms)	6.481	6.754	2.807	4.574	7.540	8.041

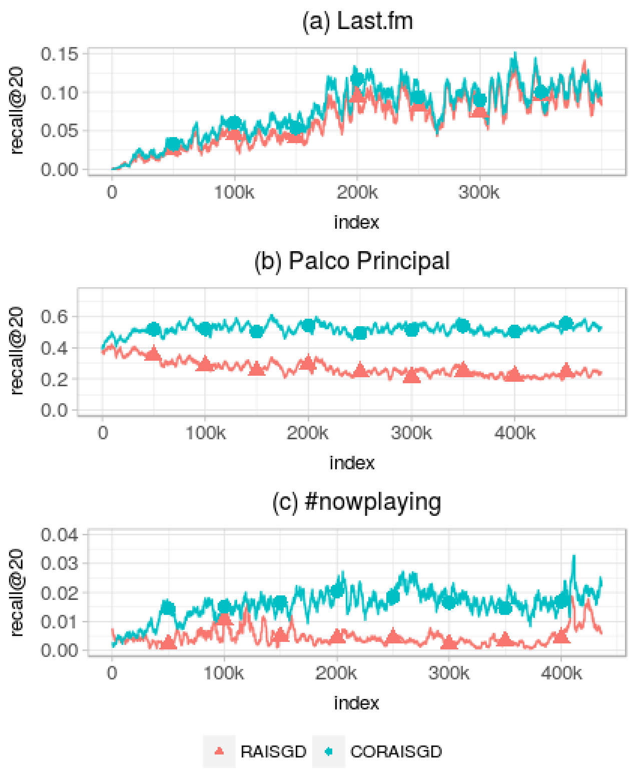

Figure 1: Moving average of recall@20 of RAISGD and CORAISGD with window size $n = 4000$.

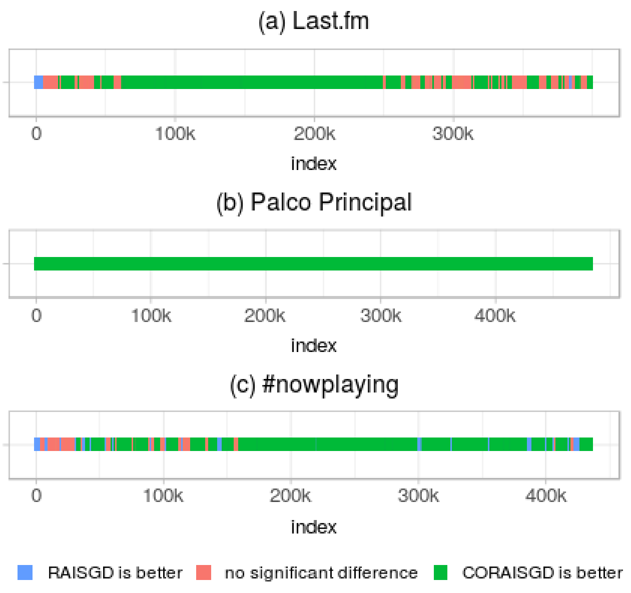

Figure 2: McNemar test's results.

6 CONCLUSIONS AND FUTURE WORK

In this work, we propose an incremental matrix co-factorization algorithm for implicit feedback. We assess the improvement in the recommendation obtained after considering additional feature dimensions in three different datasets of the music domain. For these datasets, we present the overall, as well as over time results, using prequential evaluation. We have found statistically significant improvements of our proposed algorithm over the baseline.

For future work, we will research other comparable algorithms such as MAST [22], that will need to be adapted to the recommendation task. Furthermore, we expect to study incremental ways of hyper-parameter calibration in order to decrease the effort of performing manual tuning.

ACKNOWLEDGMENTS

The work of Susan Anyosa and João Vinagre is supported by the European Regional Development Fund (ERDF) through the Operational Programme of Competitiveness and Internationalization – COMPETE 2020 – of Portugal 2020 within project POCI-01-0145-FEDER-006961, and by national funds through the Portuguese Foundation for Science and Technology (FCT) as part of project UID/EEA/50014/2013. The work of Alípio Jorge is supported by the ERDF through COMPETE 2020 within project PushNews (POCI-01-0247-FEDER-0024257).

REFERENCES

[1] Robert M. Bell and Yehuda Koren. 2007. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*. IEEE, 43–52.
[2] Òscar Celma. 2010. *Music Recommendation and Discovery - The Long Tail, Long Fail, and Long Play in the Digital Music Space.* Springer. I–XVI, 1–194 pages.
[3] Badrish Chandramouli, Justin J Levandoski, Ahmed Eldawy, and Mohamed F Mokbel. 2011. StreamRec: a real-time recommender system. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. ACM, 1243–1246.
[4] Tianqi Chen, Zhao Zheng, Qiuxia Lu, Weinan Zhang, and Yong Yu. 2011. *Feature-based matrix factorization.* Technical Report. Apex Data & Knowledge Management Lab, Shanghai Jiao Tong University.

[5] Marcos Aurélio Domingues, Alípio Mário Jorge, and Carlos Soares. 2013. Dimensions as virtual items: Improving the predictive ability of top-n recommender systems. *Information Processing & Management* 49, 3 (2013), 698–720.

[6] Yi Fang and Luo Si. 2011. Matrix Co-factorization for Recommendation with Rich Side Information and Implicit Feedback. In *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec '11)*. ACM, New York, NY, USA, 65–69.

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

[8] Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis. 2011. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 69–77.

[9] Balázs Hidasi and Domonkos Tikk. 2012. Fast ALS-based tensor factorization for context-aware recommendation from implicit feedback. *Machine Learning and Knowledge Discovery in Databases* (2012), 67–82.

[10] Liangjie Hong, Aziz S Doumith, and Brian D Davison. 2013. Co-factorization machines: modeling user interests and predicting individual decisions in twitter. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 557–566.

[11] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 263–272.

[12] Yanxiang Huang, Bin Cui, Wenyu Zhang, Jie Jiang, and Ying Xu. 2015. Tencentrec: Real-time stream recommendation in practice. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 227–238.

[13] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. 2010. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 79–86.

[14] Hiroyuki Kasai. 2016. Online low-rank tensor subspace tracking from incomplete data by CP decomposition using recursive least squares. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2519–2523.

[15] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

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

[17] Róbert Pálovics, András A. Benczúr, Levente Kocsis, Tamás Kiss, and Erzsébet Frigó. 2014. Exploiting temporal influence in online recommendation. In *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*. 273–280. https://doi.org/10.1145/2645710.2645723

[18] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. IEEE, 502–511.

[19] István Pilászy, Dávid Zibriczky, and Domonkos Tikk. 2010. Fast als-based matrix factorization for explicit and implicit feedback datasets. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 71–78.

[20] B. M. Sarwar, G. Karypis, J. Konstan, and J. Riedl. 2002. Incremental SVD-Based Algorithms for Highly Scalable Recommender Systems. In *Fifth International Conference on Computer and Information Technology*. 27–28.

[21] Ajit P. Singh and Geoffrey J. Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 650–658.

[22] Qingquan Song, Xiao Huang, Hancheng Ge, James Caverlee, and Xia Hu. 2017. Multi-aspect streaming tensor completion. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 435–443.

[23] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. 2009. Scalable collaborative filtering approaches for large recommender systems. *Journal of machine learning research* 10, Mar (2009), 623–656.

[24] João Vinagre, Alípio Mário Jorge, and João Gama. 2014. 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*.

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

[26] João Vinagre, Alípio Mário Jorge, and João Gama. 2015. Collaborative Filtering with Recency-based Negative Feedback. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC '15)*. ACM, New York, NY, USA, 963–965.

[27] Hongzhi Yin, Bin Cui, Xiaofang Zhou, Weiqing Wang, Zi Huang, and Shazia Sadiq. 2016. Joint modeling of user check-in behaviors for real-time point-of-interest recommendation. *ACM Transactions on Information Systems (TOIS)* 35, 2 (2016), 11.

[28] Eva Zangerle, Martin Pichl, Wolfgang Gassler, and Günther Specht. 2014. # nowplaying music dataset: Extracting listening behavior from twitter. In *Proceedings of the First International Workshop on Internet-Scale Multimedia Management*. ACM, 21–26.

[29] Yu Zheng and Xing Xie. 2011. Learning travel recommendations from user-generated GPS traces. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2, 1 (2011), 2.

[30] Shuo Zhou, Nguyen Xuan Vinh, James Bailey, Yunzhe Jia, and Ian Davidson. 2016. Accelerating online cp decompositions for higher order tensors. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1375–1384.