Factorization models for context-aware recommendations

Balázs Hidasi

Abstract—**The field of implicit feedback based recommender algorithms have gained increased interest in the last few years, driven by the need of many practical applications where no explicit feedback is available. The main difficulty of this recommendation task is the lack of information on the negative preferences of the users that may lead to inaccurate recommendations and scalability issues. In this paper, we adopt the use of context-awareness to improve the accuracy of implicit models—a model extension technique that was applied successfully for explicit algorithms. We present a modified version of the iTALS algorithm (coined iTALSx) that uses a different underlying factorization model. We explore the key differences between these approaches and conduct experiments on five data sets to experimentally determine the advantages of the underlying models. We show that iTALSx outperforms the other method on sparser data sets and is able to model complex user–item relations with fewer factors.**

Index Terms—**context-awareness, implicit feedback, model comparison, recommender systems, tensor factorization.**

I. Introduction

Recommender systems are information filtering tools that help users in information overload to find interesting items. For modeling user preferences, classical approaches either use item metadata (content based filtering, CBF; [1]), or user–item interactions (collaborative filtering, CF; [2]). CF algorithms proved to be more accurate than CBF methods, if sufficient interaction data (or *events*) is available [3]. CF algorithms can be further divided into memory and model based algorithms. An important subclass of the latter is the factorization algorithms (e.g. matrix factorization).

Latent factor based CF methods gained popularity due to their attractive accuracy and scalability [4]. They intend to capture user preferences by uncovering latent features that explain the observed user–item events (ratings). Models are created by the factorization of the partially observed user–item rating matrix, and the user preferences are approximated by the scalar product of the user and item factors. Matrix factorization (MF) methods may differ in the learning method and the objective function. For learning, MF methods may apply, e.g., alternating least squares (ALS; [5]), stochastic gradient [6], or a probabilistic framework [7].

Depending on the nature of the user–item interactions, recommendation problems can be classified into explicit and implicit feedback based problems. In the former case, users provide explicit information on their preferences, typically in form of ratings. In the latter case, user preferences are captured seamlessly via user activity. Implicit feedback algorithms use user interactions like viewing and purchasing retrieved e.g. from website usage logs. Obviously, implicit feedback data is less reliable because the presence of an action is only an uncertain implication that the user likes the item and the absence of an action rarely means negative preference.

Manuscript received August 28, 2014, revised December 10, 2014. The author is affiliated with Gravity Research and Development Inc. and with the Budapest University of Technology and Economics. Email: balazs.hidasi@gravityrd.com.

The implicit problem is much more important in practical applications than the explicit one, because most of the users of online e-commerce shops or services do not tend to rate items even if such an option is available[8], because (1) when purchasing they have no information on their satisfaction rate (2) they are not motivated to return later to the system to do so. In such a case, user preferences can only be inferred by interpreting user actions (also called *events*). For instance, a recommender system may consider the navigation to a particular item page as an implicit sign of preference for the item [9]. The user history specific to items are thus considered as implicit feedback on the user's taste. Note that the interpretation of implicit feedback data may not necessarily reflect user preferences which makes the implicit feedback based preference modeling a much harder task. For instance, a purchased item could be disappointing for the user, so it might not mean a positive feedback. The strength of the events' indication of preferences varies on a type by type basis. E.g. purchasing an item is a stronger indicator than looking at a product page (browsing). Missing navigational or purchase information can not be interpreted as negative feedback. The absence of the negative feedback forces us to use the information stored in the "missing" events. Most (explicit) algorithms iterate over the known ratings and use gradient descent to minimize the error function. This is not applicable in the implicit case as the number of known ratings is equal to all possible ratings as we should use the "missing" events as well. Although the explicit feedback problem is much thoroughly studied research topic, in the last few years implicit feedback algorithms have gained increased interest thanks to its practical importance; see [8], [10], [11].

Classical collaborative filtering methods only consider direct user–item interaction data to create the model. However, we may have additional information related to items, users or events, which are together termed *contextual information*. Context can be, for instance, the time or location of recommendation. Any additional information to the user–item interaction can be considered as context. Here we assume that the context dimensions are event contexts, meaning that their value is not determined solely by the user or the item; rather it is bound to the transaction itself. E.g. the time of the event is an event context, while the genres of the item is not. Integrating context into the recommender model improves the model capacity and increases accuracy, and became therefore a popular approach for the explicit algorithms recently. We argue that implicit algorithms can benefit *even more* from the context due to the uncertainty of the user feedback.

Context-aware recommendation algorithms can be divided into three groups [12]: (1) pre-filtering approaches partition the training data according to the value of the context(s) and train traditional (non context-aware) algorithms on said partitions; (2) post-filtering approaches disregard the context during training, but modify the list of recommendations according to the actual context-state; (3) contextual modeling approaches consider the context dimension(s) during the learning process.

In this paper, we use the contextual modeling approach. More specifically we extend factorization methods with context dimensions. To incorporate the context in factorization methods, the underlying model needs to be modified. However the model can be modified in several ways and each of these implies a conceptually different view on the role of the context. Building on our previous work, we present a variant of the (context-aware) iTALS algorithm [13] – coined iTALSx[1] – that uses a different underlying model.

The rest of the paper is organized as follows. Section II gives a brief overview on context-aware recommender systems. The iTALSx method is presented in Section III. The key conceptual differences between iTALS and iTALSx are highlighted in Section IV. The experimental comparison of the two approaches (conducted on five implicit feedback data sets) are described in Section V. Finally, Section VI concludes this work.

A. Notation

We will use the following notation in the rest of this paper:

- $A \circ B \circ \ldots$: The Hadamard (elementwise) product of A, B, $\ldots$. The operands are of equal size, and the result's size is also the same. The element of the result at index $(i, j, k, \ldots)$ is the product of the element of A, B, $\ldots$ at index $(i, j, k, \ldots)$.
- A_i: The i^{th} column of matrix A.
- $A_{i_1,i_2,\ldots}$: The $(i_1, i_2, \ldots)$ element of tensor/matrix A.
- K: The number of features, the main parameter of the factorization.
- D: The number of dimensions of the tensor.
- T: A D dimensional tensor that contains only zeroes and ones (preference tensor).
- W: A tensor with the same size as T (weight/confidence tensor).
- $S_{<X>}$: The size of T in dimension X (e.g. $< X >= U$ (Users)).
- N^+: The number of ratings (explicit case); non-zero elements in tensor T (implicit case).
- U, I, C: A $K \times S_{<X>}$ sized matrices. Its columns are the feature vectors for the entities in the user/item/context dimension.
- R: Training data that contains (u, i, c) triplets i.e. user–item–contex-state combinations.

[1]ITALSx is cited in some of our works as it was described in a closed (publicly non-available) technical report[14] earlier.

II. Context-aware recommender systems

Context-aware recommender systems (CARS) [15] emerged as an important research topic in the last years. Recently, entire workshops were devoted to this topic on major conferences (CARS series started in 2009 [16], CAMRa in 2010 [17]).

As we discussed earlier, pre- and post-filtering approaches use traditional recommender algorithms with some kind of filtering or splitting to consider context during learning and/or recommendation. On the other hand contextual modeling focuses on designing algorithms that incorporate context into the model itself. Tensor factorization (TF) follows the contextual modeling flavor of CARS, when contextual information (or simply: context) is incorporated into the recommendation model [12]. TF is a natural extension of matrix factorization for more dimensions, although it is not straightforward how to make it work efficiently.

Let we have a set of items, users and ratings (or events) and assume that additional contexts are available on ratings (e.g. the time of the rating). If we have N_C different contexts we can structure the ratings into a $D = N_C + 2$ dimensional tensor T. The first dimension corresponds to users, the second to items and the subsequent N_C dimensions $[3, \ldots, N_C + 2]$ are devoted to context. Note that in order to be able to use this approach, every context dimension must consists of possible context-states that are atomic and categorical values. In other words, the value of a context variable comes from a finite set of atomic values (these values are termed context-states). We want to decompose tensor T into lower rank matrices and/or tensors in a way that the reconstruction of the original tensor from its decomposition approximates the original tensor sufficiently well.

In [18] a sparse HOSVD method is presented that decomposes a D dimensional sparse tensor into D matrices and a D dimensional tensor. The authors use gradient descent on the known ratings to find the decomposition, and by doing so the complexity of one iteration of their algorithm scales *linearly* with the number of non-missing values in the original tensor and *cubically* with the number of features (K). Rendle *et al* proposed a tensor factorization method for tag recommendation [19] that was later adapted to context-aware recommendations [20]. Their model is similar to the one we use to model the relation between users, items and context states. For every entity in every dimension they use two feature vectors and the preference of user u on tag t for item i is approximated by the sum of three scalar products: (1) first user feature vector with first tag vector, (2) second user vector with second item vector and (3) first item vector with second tag vector. The second scalar product can be omitted during recommendations, since it has no effect on the ranking of tags in a given user–item relation, however it can filter noise during the training. The model is generalized to context-aware recommendations by replacing tags by items and items by context. They use gradient descent to minimize the loss function.

Our method also approximates the preferences with the sum of three scalar products, but there are major differences from the previously presented method: (1) there is only one feature vector for each entity in our model; (2) our algorithm is able to efficiently handle the implicit feedback case by using computationally effective learning scheme; (3) we use ALS to minimize the error.

The closest to our approach is our previously proposed implicit context aware tensor model [13] that approximates a given cell of the tensor as the sum of the elements in the Hadamard product of three feature vectors (i.e. it uses a full three-way model). iTALS and iTALSx uses the same loss function and optimization procedure, but they use different models and thus require different steps in the learning process. We show how the model differences affect usability in section V.

III. The iTALSx algorithm

In this section we present our context aware model and an efficient ALS (Alternating Least Squares) based method for training. ALS training fixes all but one feature matrices and computes the columns on the non fixed one by solving a least squares problem for each column.

The presented model uses one context dimension. Problems with several context dimensions can be transformed to use a single dimension by using the Descartes product of the possible context-states of each dimension. Also, the model can be extended to handle arbitrary number of dimensions, by simply adding more dot products to it.

T is a tensor that contains zeroes and ones. The number of ones in T is much lower than the number of zeroes. W contains weights to each element of T. An element of W is greater than 1 if the corresponding element if T is non-zero and 1 otherwise. This approach assumes that the presence of an event is positive preference with a high confidence while its absence is negative preference with a very low confidence. This approach is commonly used for handling implicit feedback [8][13]. In our model we decompose T into three low rank matrices (U, I and C) and approximate a given element by a sum of three dot products. The vectors used in the scalar products are the columns of the matrices corresponding to the given item/user/context state. The following equation describes the preference model:

$$\hat{T}_{u,i,c} = (U_u)^T I_i + (U_u)^T C_c + (I_i)^T C_c \tag{1}$$

During the training of the model we want to minimize the following loss function (weighted RMSE):

$$L(P,Q,C) = \sum_{u=1,i=1,c=1}^{S_U,S_I,S_C} W_{u,i,c}\left(T_{u,i,c} - \hat{T}_{u,i,c}\right)^2 \tag{2}$$

If all but one matrix are fixed (say U and C), L is convex in the non-fixed variables (elements of I in this case). The minimum of L (in I) is reached where its derivative with respect to I is zero. The columns of I can be computed separately because the derivative of L (with respect to I) is linear in I. The derivative for the i^{th} column of I is as follows:

$$\begin{aligned}\frac{\partial L}{\partial I_i} &= \underbrace{-2\sum_{u=1,c=1}^{S_U,S_C} W_{u,i,c}T_{u,i,c}\left(U_u + C_c\right)}_{\mathcal{O}} + \\ &+\underbrace{2\sum_{u=1,c=1}^{S_U,S_C} W_{u,i,c}\left(U_u + C_c\right)\left(U_u + C_c\right)^T I_i}_{\mathcal{I}} + \\ &+\underbrace{2\sum_{u=1,c=1}^{S_U,S_C} W_{u,i,c}(C_c)^T U_u\left(U_u + C_c\right)}_{\mathcal{B}} = \\ &= -2\mathcal{O} + 2\mathcal{I}I_i + 2\mathcal{B}\end{aligned} \tag{3}$$

$\mathcal{O}$ can be computed efficiently (see section III-A), but the naive computation of $\mathcal{I}$ and $\mathcal{B}$ is expensive. Therefore these expressions are further transformed by introducing $W'_{u,i,c} = W_{u,i,c} - 1$:

$$\begin{aligned}\mathcal{I} &= \underbrace{\sum_{u=1,c=1}^{S_U,S_C} W'_{u,i,c}\left(U_u + C_c\right)\left(U_u + C_c\right)^T}_{\mathcal{I}_1} + \\ &+\underbrace{\sum_{u=1,c=1}^{S_U,S_C}\left(U_u + C_c\right)\left(U_u + C_c\right)^T}_{\mathcal{I}_2}\end{aligned} \tag{4}$$

The sum in $\mathcal{I}_1$ contains at most N^+ non-zero members, because $W'_{u,i,c}$ is zero if the corresponding element is T is zero. Thus its computation is efficient. $\mathcal{I}_2$ is independent of i (it is the same for each column of Q) thus can be precomputed. However its naive computation is still expensive, therefore we further transform $\mathcal{I}_2$ as follows:

$$\begin{aligned}\mathcal{I}_2 &= S_C\underbrace{\sum_{u=1}^{S_U} U_u\left(U_u\right)^T}_{\mathcal{M}^{(U)}} + S_U\underbrace{\sum_{c=1}^{S_C} C_c\left(C_c\right)^T}_{\mathcal{M}^{(C)}} + \\ &+\underbrace{\left(\sum_{c=1}^{S_C} C_c\right)}_{\mathcal{X}^{(C)}}\underbrace{\left(\sum_{u=1}^{S_U} U_u\right)^T}_{(\mathcal{X}^{(U)})^T} + \underbrace{\left(\sum_{u=1}^{S_U} U_u\right)}_{\mathcal{X}^{(U)}}\underbrace{\left(\sum_{c=1}^{S_C} C_c\right)^T}_{(\mathcal{X}^{(C)})^T} = \\ &= S_C\mathcal{M}^{(U)} + S_U\mathcal{M}^{(C)} + \mathcal{X}^{(C)}(\mathcal{X}^{(U)})^T + \mathcal{X}^{(U)}(\mathcal{X}^{(C)})^T\end{aligned} \tag{5}$$

The members ($\mathcal{M}^{(U)}$, $\mathcal{M}^{(C)}$, $\mathcal{X}^{(U)}$, $\mathcal{X}^{(C)}$) in equation 7 can be computed efficiently. Note that the recomputation of $\mathcal{M}^{(U)}$ and $\mathcal{X}^{(U)}$ is only necessary when U changes and therefore we include the cost of recomputing these variables to the cost of recomputing U. We can perform similar steps for $\mathcal{B}$, separating it into two parts, one of which can be rewritten using the variables above. The decomposition is shown in the following equation:

$$\mathcal{B} = \underbrace{\sum_{u=1,c=1}^{S_U,S_C} W'_{u,i,c}(C_c)^T U_u \left(U_u + C_c\right) +}_{\mathcal{B}_1} \\ + \mathcal{M}^{(U)}\mathcal{X}^{(C)} + \mathcal{M}^{(C)}\mathcal{X}^{(U)} \tag{6}$$

Now all we have to do in order compute the desired column of I is to solve a $K \times K$ system of linear equations:

$$I_i = \left(\mathcal{I}_2 + \mathcal{I}_1\right)^{-1}\left(\mathcal{O} - \left(\mathcal{B}_2 + \mathcal{B}_1\right)\right) \tag{7}$$

Bias and regularization can be easily added to the method, thus they are omitted in this deduction for the sake of clearer presentation.

Algorithm III.1 presents the pseudocode for training the model, that is the straight translation of the deduction above. The method is named iTALSx.

A. Complexity

The complexity of one epoch (i.e. computing each matrix once) is $O\left(K^3(S_U + S_I + S_C) + K^2N^+\right)$, thus it scales linearly with the number of non-zeros in the tensor and cubically with the number of factors. Since in practical problems $N^+ \gg S_U + S_I + S_C$, the scaling of the method is practically quadratical in K when small $(10 \dots 400)$ K values are used (also common in practice).[2]

The complexity of recomputing I (as in the deduction above) is $O(K^3 S_I + K^2 N^+)$. This complexity also contains the recomputation of $\mathcal{M}^{(I)}$ and $\mathcal{X}^{(I)}$ that are needed later for the computation of the other two matrices. The aforementioned complexity consists of calculating

- (1) $\mathcal{O}$ for each column in equation (3) in $O(KN^+)$ time as only N_i^+ elements of T are non-zeroes for i^{th} item ($N^+ = \sum_{i=1}^{M} N_i^+$);
- (2) $\mathcal{I}_1$ for each column in equation (4) in $O(K^2N^+)$ time as $W'_{u,i,c} = (W_{u,i,c} - 1)$ is zero if the value of $T_{u,i,c}$ is zero;
- (3) $\mathcal{I}_2$ in equation (7) in $O(K^2)$ time from the precomputed values $\mathcal{M}^{(U)}$, $\mathcal{M}^{(C)}$, $\mathcal{X}^{(U)}$, $\mathcal{X}^{(C)}$;
- (4) $\mathcal{B}$ in equation (6) in $O(KN^+ + K^2)$ time analogously to the computation of $\mathcal{I} = \mathcal{I}_1 + \mathcal{I}_2$;
- (5) solving the systems of equations for all columns in $O(S_I K^3)$ time;
- (6) recomputing $\mathcal{M}^{(I)}$ and $\mathcal{X}^{(I)}$ based on the new I matrix in $O(S_I K^2)$ time

IV. Comparison with iTALS

In earlier work we recently proposed a context aware tensor model (coined iTALS) for the implicit feedback problem. This model is a full three-way model that approximates the elements in T with the sum of the elements in the Hadamard products of three vectors. Mathematically the model of iTALS [13] contains the iTALSx model as:

$$\begin{aligned}
\hat{T}^{\text{iTALS}}_{u,i,c} &= 1^T\left(U_u \circ I_i \circ C_c\right) \\
3 \cdot \hat{T}^{\text{iTALS}}_{u,i,c} &= 1^T\left(U_u \circ I_i \circ C_c\right) + 1^T\left(U_u \circ I_i \circ C_c\right) + \\
&\quad + 1^T\left(U_u \circ I_i \circ C_c\right) \\
\hat{T}^{\text{iTALSx}}_{u,i,c} &= 1^T\left(I_i \circ C_c\right) + 1^T\left(U_u \circ C_c\right) + 1^T\left(U_u \circ I_i\right) \\
\hat{T}^{\text{iTALSx}}_{u,i,c} &= 1^T\left(1 \circ I_i \circ C_c\right) + 1^T\left(U_u \circ 1 \circ C_c\right) + \\
&\quad + 1^T\left(U_u \circ I_i \circ 1\right),
\end{aligned} \tag{8}$$

where $\circ$ denotes the Hadamard product of the argument vectors.

It is more interesting to compare the models from the recommendation aspect. The main goal of a collaborative

Algorithm III.1 iTALSx algorithm

Input: T: $S_1 \times S_2 \times S_3$ sized tensor of zeroes and ones; W: $S_1 \times S_2 \times S_3$ sized tensor containing the weights; K: number of features; E: number of epochs; $\{\lambda_m\}_{m=1,2,3}$: regularization parameters
Output: $\{M^{(i)}\}_{m=1,2,3}$ $K \times S_i$ sized matrices
Note: $S_1 = S_U$, $S_2 = S_I$, $S_3 = S_C$, $M^{(1)} = U$, $M^{(2)} = I$, $M^{(3)} = C$

```
procedure iTALSx($T, W, K, E$)
 1: for $m = 1, \dots, 3$ do
 2:     $M^{(m)} \leftarrow$ Random $K \times S_m$ sized matrix
 3:     $\mathcal{M}^{(m)} \leftarrow \sum_{j=1}^{S_m} M_j^{(m)}(M_j^{(m)})^T$
 4:     $\mathcal{X}^{(m)} \leftarrow \sum_{j=1}^{S_m} M_j^{(m)}$
 5: end for
 6: for $e = 1, \dots, E$ do
 7:     for $m = 1 \dots, 3$ do
 8:         $p \leftarrow (m-1)\%3$
 9:         $n \leftarrow (m+1)\%3$
10:         $\mathcal{I}_2 \leftarrow \mathcal{M}^{(p)}S_n + \mathcal{M}^{(n)}S_p + \mathcal{X}^{(p)}(\mathcal{X}^{(n)})^T + \mathcal{X}^{(n)}(\mathcal{X}^{(p)})^T$
11:         $\mathcal{B}_2 \leftarrow \mathcal{M}^{(p)}\mathcal{X}^{(p)} + \mathcal{M}^{(n)}\mathcal{X}^{(p)}$
12:         for $i = 1..S_m$ do
13:             $\mathcal{I} \leftarrow \mathcal{I}_2$
14:             $\mathcal{O} \leftarrow 0$
15:             $\mathcal{B} \leftarrow \mathcal{B}_2$
16:             for all $\{t | t = T_{j_1,j_2,j_3}, j_m = i, t \neq 0\}$ do
17:                 $w \leftarrow$ corresponding value in $W$ to $t$ in $T$
18:                 $v \leftarrow M^{(p)} + M^{(n)}$
19:                 $\mathcal{I} \leftarrow \mathcal{I} + wvv^T$
20:                 $\mathcal{O} \leftarrow \mathcal{O} + wtv$
21:                 $\mathcal{B} \leftarrow \mathcal{B} + wM^{(p)}(M^{(n)})^T v$
22:             end for
23:             $M_i^{(m)} \leftarrow (\mathcal{I} + \lambda_m I)^{-1}(\mathcal{O} - \mathcal{B})$
24:         end for
25:         $\mathcal{M}^{(m)} \leftarrow \sum_{j=1}^{S_m} M_j^{(m)}(M_j^{(m)})^T$
26:         $\mathcal{X}^{(m)} \leftarrow \sum_{j=1}^{S_m} M_j^{(m)}$
27:     end for
28: end for
29: return $\{M^{(m)}\}_{m=1\dots3}$
end procedure
```

[2]This can be reduced to a theoretically quadratic and practically linear scaling by applying approximate least squares solvers like conjugate gradient instead of the exact solver, like in [21].

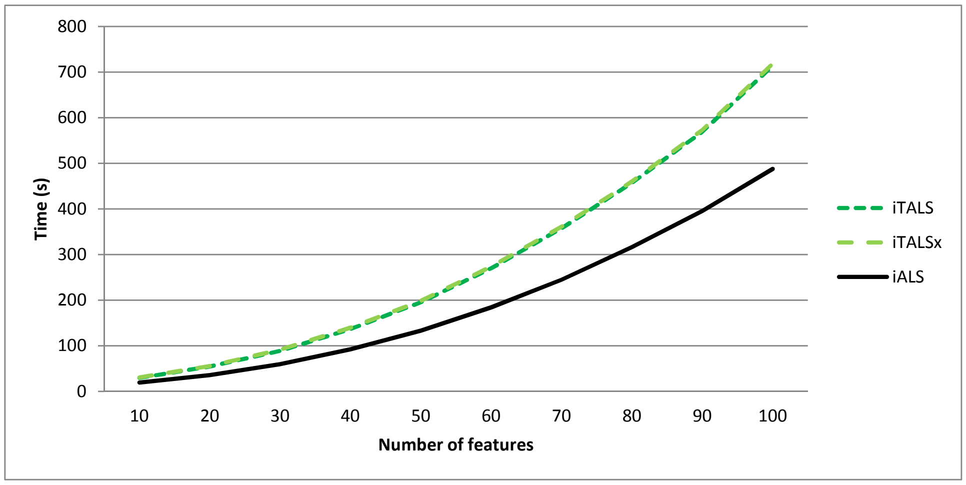

Fig. 1. The time of one epoch (computing each feature matrix once) with iTALS and iTALSx using different number of features. (Measurements on the VoD data set; only one core used.) Results for iALS are also depicted.

filtering algorithm is to learn the user–item relations (e.g. which user likes which item). iTALS adds context to the model and approximates the user–item relation in the 3 dimensional space. It reweights the user–item relations by a context state dependent vector, which becomes more accurate with more factors [13]. On the other hand, iTALSx uses a composite model and approximates the user–item relation by the sum of approximations in user–item, user–context and item–context sub-spaces, where the feature vectors in the sub-spaces are constrained by requiring a single feature vector for each entity. Consequently, the descriptive power of iTALS is larger, which can be however only leveraged at a sufficiently fine resolution of the feature space, requiring many factors. At low factor models, the boundaries of different characteristics is blurred by reweighting and the model becomes less precise. In such cases, iTALSx is expected to be more accurate, since the sub-space models can be learnt easier.

A. *Complexity and training times*

The complexity is $O(N^+K^2 + (S_U + S_I + S_C)K^3)$ for both iTALS and iTALSx. Since in practice $N^+ \gg (S_U + S_I + S_C)$, each method scales with K^2 when low-factor models are used. However the training time of iTALSx is slightly higher, because (a) iTALS does not require $\mathcal{X}^{(m)}$ for its computations; (b) the computations in iTALSx require a few extra operations (see Figure 1).

Figure 1 also contains the training times for non-context-aware (2D) iALS algorithm, that uses a similar method for learning. The complexity of iALS is $O(N^+K^2 + (S_U + S_I)K^3)$. This means that the running times of iALS and the context-aware methods differ only in a constant multiplier, that is proportional to the number of matrices to be recomputed (see Figure 1), but the time to compute one feature matrix is virtually the same for these algorithms.

V. Results

We used five data sets to evaluate our algorithm. Two of them (Grocery and VoD) are proprietary data sets and contain real-life implicit data. The other three data sets (LastFM [22], TV1 and TV2 [23]) are publicly available, but might have been transformed/cleaned prior release. The properties of the data sets are summarized in Table I. The column "Multi" shows the average multiplicity of user-item pairs in the training events.[3] Data density is measured without context, with seasonality (-S) and with sequentiality (-Q). The first event in the test data is after the last event of the training data. The length of the test period was selected to be at least one day, and depends on the domain and the frequency of events. We used the artists as items in LastFM.

The evaluation metric used here is recall@N. Recall is the proportion of the number of recommended and relevant items to the number of relevant items. Item i is considered relevant for user u if the user has at least one event for that item in the test set. The item is recommended at cut-off N if it belongs to the topN of the recommendation list[4]. We chose cut-off 20 because the length of the recommendation list is limited as users are exposed to a few recommended items at a time. The evaluation is event based, meaning that if the user has multiple events on an item in the test set then that item is considered multiple times.

Recall@N suites the recommendation task really well from a practical point of view. During a session the user is exposed to some recommended items (e.g. a few on each page visited) and the recommendation is successful if she interacts (e.g. buys, watches) with these items. The items further down the recommendation list are irrelevant, because they won't be shown to the user. 20 as cut-off is a fair estimation of the items the user sees during a session (e.g. 4 pages visited, 5 recommendations per page). In most practical settings the order of the topN items is irrelevant due to the placement of

[3]This value is 1.0 at two data sets: TV1 and TV2 due to possible filtering of duplicate events.

[4]The recommendation list is generated by ordering the items for a user (under the given context) by their predicted preference values in descending order.

TABLE I
MAIN PROPERTIES OF THE DATA SETS

Dataset	Domain	Training set							Test set	
		#Users	#Items	#Events	Density	Density-S	Density-Q	Multi	#Events	Length
Grocery	E-grocery	24947	16883	6238269	0.61%	0.15%	9.37E-5%	3.0279	56449	1 month
TV1	IPTV	70771	773	544947	1.02%	0.17%	1.63E-3%	1.0000	12296	1 week
TV2	IPTV	449684	3398	2528215	0.17%	0.028%	6.42E-5%	1.0000	21866	1 day
LastFM	Music	992	174091	18908597	0.52%	0.21%	2.31E-5%	21.2715	17941	1 day
VoD	IPTV	480016	46745	22515406	0.25%	0.046%	1.91E-5%	1.2135	1084297	1 day

recommendations on the site.[5]

We experimented with two types of context. The first is seasonality. It consists of a season (or periodicity) and time bands therein. Each event is assigned to a time band based on its time stamp. The idea behind seasonality is that people have daily/weekly/yearly routines and those are different for different types of people (same goes for the items' consumption patterns). As season, we define *one week* for Grocery (as people usually go shopping once a week) and *one day* for the other data sets (as movie and music consumption shows daily periodicity). The days of the week were used for Grocery and four hour periods within the day for the other data sets as time bands.

The second type of context is sequentiality [13]. The context state of an event is the previously consumed item of the same user. This context enables distinction between item groups with different repetitiveness patterns (e.g. item groups that can and those that should not be recommended subsequently). Sequentiality tackles this problem through the co-occurrence of the items. It – implicitly – also serves weak information on the user or her properties (e.g.: mood) if the subsequent events are close to each other. Each context state corresponds to a singular preceding item.

iTALSx is compared mainly to iTALS in order to find the differences between the behavior of the pairwise model and the three-way model. Results for the non-context-aware iALS [8] are also presented as a baseline. The number of features was set to 20, 40 and 80, the number of epochs was 10. These are typical settings in real life environments. Other parameters such as regularization coefficients were optimized on a hold-out set of the training data, then the algorithm was retrained with the optimal parameters on the whole training data.

Table II contains the results. Measurements with seasonality and sequentiality are denoted with the -S and -Q postfix respectively. As expected, context improves recommendation accuracy. There are two contradictory examples where the context-unaware method performs significantly better than iTALS. This is due to sensitivity of the elementwise model to noise and the poor quality of this seasonal context for the TV2 dataset and the outstanding sparsity of TV2 dataset compared to the others in this setting. The range of improvement for iTALS and iTALSx over the context-unaware baseline is $11\% - 53\%$ and $7\% - 63\%$ respectively, with seasonality; $7\% - 248\%$ and $11\% - 274\%$ with sequentiality. I.e. iTALSx increases the accuracy slightly more than iTALS.

[5]This does not apply if some items are highlighted from the recommendations, e.g. the picture for the first recommended item is larger.

TABLE II
RECALL@20 FOR iTALS, iTALSx AND iALS. MEASUREMENTS WITH SEASONALITY AND SEQUENTIALITY ARE DENOTED WITH THE -S AND -Q POSTFIX RESPECTIVELY.

K	iALS	iTALS-S	iTALSx-S	iTALS-Q	iTALSx-Q
GROCERY					
20	0.0649	0.0990	0.1027	0.1220	0.1182
40	0.0714	0.1071	0.1164	0.1339	0.1299
80	0.0861	0.1146	0.1406	0.1439	0.1431
TV1					
20	0.1189	0.1167	0.1248	0.1417	0.1524
40	0.1111	0.1235	0.1127	0.1515	0.1417
80	0.0926	0.1167	0.0942	0.1553	0.1295
TV2					
20	0.2162	0.1734	0.2220	0.2322	0.2393
40	0.2161	0.2001	0.2312	0.3103	0.2866
80	0.2145	0.2123	0.2223	0.2957	0.3006
LASTFM					
20	0.0448	0.0674	0.0503	0.1556	0.1675
40	0.0623	0.0888	0.0599	0.1657	0.1869
80	0.0922	0.1290	0.0928	0.1864	0.1984
VOD					
20	0.0633	0.0778	0.0790	0.1039	0.0821
40	0.0758	0.0909	0.0916	0.1380	0.1068
80	0.0884	0.0996	0.0990	0.1723	0.1342

The better between iTALS and iTALSx in the same setting (i.e.: same context, number of features, dataset) is highlighted by a light gray background. Generally, iTALSx performs better if the number of features is lower. Also, there seems to be a loose connection between the density of the dataset and the relative accuracy of the two models. With a given context, iTALSx performs better if the density of the dataset is lower. High sparsity (lower density) is a common property of real life datasets, therefore iTALSx is beneficial for practical applications.

Figure 2 compares iTALS and iTALSx using high number of features on the LastFM dataset. With seasonality, iTALS is already better than iTALSx, even with 40 features. The recommendation accuracy of iTALS improves faster as the

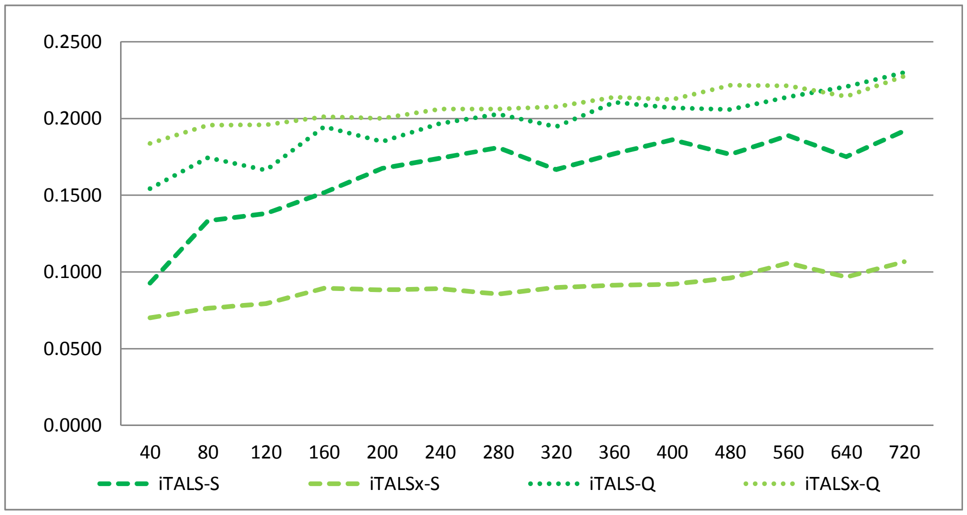

Fig. 2. Recall@20 values for iTALS and iTALSx with seasonality (-S) and sequentiality (-Q) with the number of features ranging from 40 to 720 on the LastFM dataset.

number of features increases. With sequentiality, iTALSx starts off with significantly better accuracy, but as the number of features increase, the difference becomes less significant and it disappears at high factor models. The speed of accuracy improvement is better for iTALS in both cases.

The blurring effect of the low feature models makes learning difficult for iTALS, especially if the dataset is sparse. Sparser datasets are generally more noisy, and the elementwise model is more sensitive to noise by nature, because of the reweighting of the user–item relation in that model. Our assumption about the learning capabilities of the algorithms and their connection to the finer representation of entities are underpinned as iTALS can outperform iTALSx when the number of features is sufficiently large or if the dataset is more dense. These results imply that one should use iTALSx when the dataset is sparse and we can not afford high feature models (that is most common in practical applications).

VI. Conclusion

In this paper we presented iTALSx, an efficient context-aware factorization method for implicit feedback data, which approximates preferences as the sum of three scalar products. It scales cubically (quadratically in practice) with the number of features (K) and linearly with the number of events. We compared it to iTALS, a similar method that uses a different (three-way) model. We found that both models have their advantages. The pairwise model of iTALSx is more efficient in terms of accuracy if the number of features is low and the dataset is more sparse. This is a usual setting in real life problems. However if one can afford high factor models or the data is more dense, iTALS should be used, as its learning capability is higher. Thus it can achieve better results if the number of features is sufficient.

Acknowledgements

The research leading to these results was partially performed in the CrowdRec project, which has received funding from the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement n° 610594.

References

[1] P. Lops, M. Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender Systems Handbook*. Springer, 2011, pp. 73–105.

[2] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, pp. Article ID 421 425 (1–19), 2009.

[3] I. Pilászy and D. Tikk, "Recommending new movies: Even a few ratings are more valuable than metadata," in *Recsys'09: ACM Conf. on Recommender Systems*, 2009, pp. 93–100.

[4] Y. Koren and R. Bell, "Advances in collaborative filtering," in *Recommender Systems Handbook*, F. Ricci *et al.*, Eds. Springer, 2011, pp. 145–186.

[5] R. Bell and Y. Koren, "Scalable collaborative filtering with jointly derived neighborhood interpolation weights," in *ICDM'07: IEEE Int. Conf. on Data Mining*, 2007, pp. 43–52.

[6] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Major components of the Gravity recommendation system," *SIGKDD Explor. Newsl.*, vol. 9, pp. 80–83, December 2007.

[7] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Advances in Neural Information Processing Systems 20*. MIT Press, 2008.

[8] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *ICDM-08: IEEE Int. Conf. on Data Mining*, 2008, pp. 263–272.

[9] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender Systems Handbook*. Springer US, 2011, pp. 1–35.

[10] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic, "Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering," in *Proceedings of the sixth ACM conference on Recommender systems*, ser. RecSys '12. ACM, 2012, pp. 139–146.

[11] G. Takács and D. Tikk, "Alternating least squares for personalized ranking," in *Proceedings of the sixth ACM conference on Recommender systems*, ser. RecSys '12. ACM, 2012, pp. 83–90.

[12] G. Adomavicius and A. Tuzhilin, "Context-aware recommender systems," in *Recsys'08: ACM Conf. on Recommender Systems*, 2008, pp. 335–336.

[13] B. Hidasi and D. Tikk, "Fast ALS-based tensor factorization for context-aware recommendation from implicit feedback," in *Proc. of the ECML-PKDD, Part II*, ser. LNCS. Springer, 2012, no. 7524, pp. 67–82.

[14] B. Hidasi, "Technical report on iTALSx," Gravity R&D Inc., Tech. Report Series 2012-2, 2012.

[15] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, "Incorporating contextual information in recommender systems using a multidimensional approach," *ACM Trans. Inf. Syst.*, vol. 23, no. 1, pp. 103–145, 2005.

[16] G. Adomavicius and F. Ricci, "Workshop on context-aware recommender systems (CARS-2009)," in *Recsys'09: ACM Conf. on Recommender Systems*, 2009, pp. 423–424.

[17] A. Said, S. Berkovsky, and E. W. D. Luca, "Putting things in context: Challenge on context-aware movie recommendation," in *CAMRa'10: Workshop on Context-Aware Movie Recommendation*, 2010, pp. 2–6.

[18] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering," in *Recsys'10: ACM Conf. on Recommender Systems*, 2010, pp. 79–86.

[19] S. Rendle and L. Schmidt-Thieme, "Pairwise interaction tensor factorization for personalized tag recommendation," in *WSDM'10: ACM Int. Conf. on Web Search and Data Mining*, 2010, pp. 81–90.

[20] Z. Gantner, S. Rendle, and L. Schmidt-Thieme, "Factorization models for context-/time-aware movie recommendations," in *Proc. of the Workshop on Context-Aware Movie Recommendation*, 2010, pp. 14–19.

[21] B. Hidasi and D. Tikk, "Context-aware recommendations from implicit data via scalable tensor factorization," *ArXiv e-prints*, 2013.

[22] O. Celma, *Music Recommendation and Discovery in the Long Tail*. Springer, 2010.

[23] P. Cremonesi and R. Turrin, "Analysis of cold-start recommendations in IPTV systems," in *Proc. of the 2009 ACM Conference on Recommender Systems*, 2009.

Balázs Hidasi is a datamining researcher. His research interests cover a broad spectrum of machine learning / data mining algorithms and problems. His recent research revolves around recommender algorithms for real life recommendation problems, that include research related to implicit feedback, context-awareness, hybrid collaborative filtering and so on. Before that he worked on time series classification. Since 2010 he is employed by Gravity Research and Development Inc., a recommendation service provider company. There he carries out his research and applies the resulting algorithms directly to real life problems. He graduated with highest honors from the Budapest University of Technology, and received his masters degree in computer science and engineering in 2011.