Weighted Tensor Decompositions for Context-aware Collaborative Filtering

JOEY DE PAUW, University of Antwerpen, Belgium
BART GOETHALS, University of Antwerpen, Belgium and Monash University, Australia

Over recent years it has become well accepted that user interest is not static or immutable. There are a variety of contextual factors, such as time of day, the weather or the user's mood, that influence the current interests of the user. Modelling approaches need to take these factors into account if they want to succeed at finding the most relevant content to recommend given the situation.

A popular method for context-aware recommendation is to encode context attributes as extra dimensions of the classic user-item interaction matrix, effectively turning it into a tensor, followed by applying the appropriate tensor decomposition methods to learn missing values. However, unlike with matrix factorization, where all decompositions are essentially a product of matrices, there exist many more options for decomposing tensors by combining vector, matrix and tensor products. We study the most successful decomposition methods that use weighted square loss and categorize them based on their tensor structure and regularization strategy. Additionally, we further extend the pool of methods by filling in the missing combinations.

In this paper we provide an overview of the properties of the different decomposition methods, such as their complexity, scalability, and modelling capacity. These benefits are then contrasted with the performances achieved in offline experiments to gain more insight into which method to choose depending on a specific situation and constraints.

CCS Concepts: • **Information systems** → *Learning to rank*; **Recommender systems**; *Collaborative filtering*.

1 INTRODUCTION

Collaborative filtering techniques are currently the most used and most studied approaches for top-k recommendation with implicit feedback. In collaborative filtering, a user's recommendations are based on the history of items they consumed and how similar they are to other users. This approach has its limitations as often much more data is available to learn from such as personal information of the user, metadata of the item or contextual information about the interactions. Especially the usage of contexts is found to be indispensable to model the dynamic nature of user behaviour and to make sure the recommender system can adapt to the current needs of the user [1, 19].

Implicit feedback data is often represented as a user-by-item binary interaction matrix where each interaction (view/click/…) is represented by a one and everything else as zero. As the item catalogue grows, this matrix naturally becomes increasingly sparse since every user typically only observes a limited amount of items. This problem is exacerbated further as context dimensions are added, because each dimension exponentially increases the sparsity of the tensor.

Learning from sparse, positive-only data is one of the key challenges in recommendation and requires specialised learning methods such as negative sampling or importance weighting [14, 20] to prevent models from overfitting to the missing data [8, 20]. Both approaches have their pros and cons: while negative sampling has more efficient training steps, convergence may be slow and highly dependent on the learning rate. Importance weighting on the other hand can use the whole data and is hence more costly, but it is often found to be more effective [4, 8]. In this paper, we focus on whole-data models with weighted square loss. Weighted Matrix Factorization (WMF), also called iALS [14, 20], pioneered this class of models and is still known to achieve competitive results while having highly scalable learning and prediction routines [22]. After its introduction, many extensions were proposed, among which three variants for context-aware recommender systems (CARS) [5, 10, 11], where each variant uses a different tensor decomposition method.

Table 1 classifies the WMF-based CARS models in terms of *decomposition method*, *tensor structure* and *regularization strategy*. First, for tensor structure, we distinguish between models that support arbitrary dimensions and models that are limited to three dimensions. In the latter case, if there are multiple context features, they have to be stacked in one dimension. Then, the standard regularization strategy 'zero' uses the same ℓ^2-norm for all factors, pulling them all towards the

Table 1. Overview of the studied CARS models. Methods marked with asterisk (*) are our contributions.

Tensor Structure:	**Up to 3D / Stacked to 3D**		**Multidimensional**	
Regularization:	**zero**	**one**	**zero**	**one**
CP [13]	iTALSs [11]	iTALSs one*	iTALS [11]	iTALS one*
PITF [23]	iTALSx [10]	iTALSx [10]	FTF [5]	FTF [5]
TTF [18]	WTF*	WTF one*	–	–

origin. Our new strategy 'one' regularizes the context factors towards the mathematical identity which benefits learning as per our experimental results. Finally, we consider the three most practical tensor decomposition methods: CANDECOMP/PARAFAC (CP) [13] where each interaction is modelled by a product of vectors, Pairwise Interaction Tensor Factorization (PITF) [23] where the sum of pairwise products of vectors is used, and Tensor Train Factorization (TTF) [18] where user and item factors are vectors and contexts are modelled as matrices. The Tucker decomposition [13, 26] is not included because no WMF-based CARS algorithm with this decomposition has been published yet [7], likely because the precomputation step that leads to the efficiency of WMF, is not possible due to the use of a core tensor.

Six spaces in this table are already filled by previously proposed methods and we fill in four of the missing combinations that are indicated with an asterisk. Note that iTALSx and FTF are classified both under zero and one regularization for PITF as 'zero' already is the mathematical identity for the factors and it hence coincides with the 'one' variant. Also worth mentioning is the General Factorization Framework [12] which studies combinations of CP and PITF, and as such would fit somewhere between these rows of the table. Only two entries are left for multidimensional TTF, which is out of the scope of this paper and remains open for future research. This paper has three main contributions:

(1) Introducing Weighted Tensor Factorization (WTF), a novel WMF-based model for CARS with the TTF structure.
(2) Exploring a new regularization method for context factors that aligns more with the recommendation task.
(3) Experimental evaluation and comparison of the listed models on three publicly available datasets.

2 MODELS

Let $X \in \{0,1\}^{m \times n \times l_1 \times \cdots \times l_d}$ be the binary interaction tensor and $\hat{X} \in \mathbb{R}^{m \times n \times l_1 \times \cdots \times l_d}$ be the reconstructed values given a tensor decomposition method. Then the class of models we study all share the following loss function:

$$\mathcal{L} = \left\| \sqrt{W} \odot \left(X - \hat{X} \right) \right\|_F^2 + \mathcal{R} \qquad \text{with } W = 1 + \alpha X \tag{1}$$

Where $\odot$ denotes the elementwise product and $\|\cdot\|_F$ the Frobenius norm. W are the weights, and $\mathcal{R}$ represents the regularization. Intuitively, this loss takes a weighted sum of the squared error for both seen and unseen interactions. However, all unseen interactions are given the same weight of one, which allows for efficient optimization [14]. Positive interactions are given a constant weight of $1 + \alpha$, but it can also be dependent on the strength of the signal (e.g.: rating, recency or watch duration). By using square loss, most models in this class are bi-convex and can be optimized with Alternating Least Squares (ALS) [14, 20]. In ALS, we learn the user factors while keeping the item and context factors constant and vice-versa for the others until convergence to a local optimum [21].

Notice that in Equation 1, X, W and $\hat{X}$ are tensors of dimension $2+d$ for multidimensional models, with d the amount of context dimensions. Each context feature is added as a separate dimension which allows dependencies between contexts to be modelled as well. In most practical cases however, there are no meaningful interactions between contexts to learn or there is not enough data to learn them from [11]. For those cases, we can reduce the model complexity by *stacking* all context features into a single dimension, making it so only a three-dimensional tensor needs to be factorized. For the

training procedure, nothing changes except for a single interaction that can be mapped to multiple ones in the tensor with the stacking procedure. To compute predictions we simply take the average of the predictions for every context. We also chose not to learn from missing values in our models, i.e.: there is no substitute factor that is learned for all missing values. Doing so may lead to overfitting to errors in the data that do not generalize. Instead we use a static *default factor* for missing values that depends on the regularization, namely the target towards which the context factors are being pulled.

The next sections explain the loss function, update formulas (found by setting the partial derivatives to zero) and complexity for each of the three decomposition methods. We use a uniform notation with $P \in \mathbb{R}^{m \times k}$ for the user factors, $Q \in \mathbb{R}^{n \times k}$ for item factors and $B^{(c)}$ for contexts. Implementations of the models in Python and experiments are available: https://github.com/JoeyDP/CARS-Experiments.

2.1 CP Decomposition: iTALS & iTALSs

The CP decomposition models every entry as a product of vectors, leading to the following loss and update formula:

$$\mathcal{L}^{(iTALS)} = \sum_{u}^{m} \sum_{i}^{n} \sum_{c_1}^{l_1} \cdots \sum_{c_d}^{l_d} W_{u,i,c_1,\ldots,c_d} \left(X_{u,i,c_1,\ldots,c_d} - P_u \operatorname{dm}(B^{(1)}_{c_1}) \cdots \operatorname{dm}(B^{(d)}_{c_d}) Q_i^\top \right)^2 + \lambda \left(\|P\|_F^2 + \|Q\|_F^2 + \sum_{c}^{d} \left\| B^{(c)} - \vec{1}\vec{1}^\top \right\|_F^2 \right)$$

$$P_u^\top = \left((B^{(1)\top} B^{(1)}) \odot \cdots \odot (B^{(d)\top} B^{(d)}) \odot (Q^\top Q) + \sum_{i}^{n} \sum_{c_1}^{l_1} \cdots \sum_{c_d}^{l_d} W'_{u,i,c_1,\ldots,c_d} \operatorname{dm}(B^{(1)}_{c_1}) \cdots \operatorname{dm}(B^{(d)}_{c_d}) Q_i^\top Q_i \operatorname{dm}(B^{(d)}_{c_d}) \cdots \operatorname{dm}(B^{(1)}_{c_1}) + \lambda I \right)^{-1}$$
$$\left(\sum_{i}^{n} \sum_{c_1}^{l_1} \cdots \sum_{c_d}^{l_d} W_{u,i,c_1,\ldots,c_d} X_{u,i,c_1,\ldots,c_d} \operatorname{dm}(B^{(1)}_{c_1}) \cdots \operatorname{dm}(B^{(d)}_{c_d}) Q_i^\top \right)$$

with $W_{u,i,c_1,\ldots,c_d} = W'_{u,i,c_1,\ldots,c_d} + 1$. Here only the update formula for the user factor is given, as the item and context factors are computed similarly. Notice that, despite our usage of *diagonal matrix* notation $\operatorname{dm}(\cdot)$, all factors are vectors and only their element-wise product is needed in this loss, which is commutative. Hence the derivation of all factors follows the same structure, except for one additional term in the context factors corresponding to the $-\vec{1}\vec{1}^\top$ part in their regularization. If we omit this part, all context factors are regularized towards zero (iTALS), and by including this term the context factors are pulled towards the vector of ones (iTALS one). There are two reasons for taking this approach. First, the elementwise product with a vector of ones is the identity operation in this decomposition, so if a certain context factor is not informative for predicting interactions, the model is not penalized for not learning from it. Intuitively the context factor can be seen as a learned offset from the base user-item prediction. Second, we can choose the 'default factor' for missing context values to be the vector of ones without changing the magnitude of the prediction. Indeed if all known context factors are learned to be around zero, suddenly using a vector of ones in the loss will throw off the user and item factors as the prediction will suddenly be much larger. iTALSs is the *stacked* version of iTALS. The computational complexity for one update step of all factors is $O(k^3(m + n + l_1 + \cdots + l_d) + k^2 p)$ with p the amount of interactions.

2.2 PITF Decomposition: iTALSx

In PITF all factors are also vectors. The difference is that pairwise combinations are multiplied. For 3D we get:

$$\mathcal{L}^{(iTALSx)} = \sum_{u}^{m} \sum_{i}^{n} \sum_{c}^{l} W_{u,i,c} \left(X_{u,i,c} - P_u Q_i^\top - P_u B_c^\top - Q_i B_c^\top \right)^2 + \lambda \left(\|P\|_F^2 + \|Q\|_F^2 + \|B\|_F^2 \right)$$

$$P_u^\top = \left(\sum_i^n \sum_c^l W'_{u,i,c} (Q_i + B_c)^\top (Q_i + B_c) + l \cdot Q^\top Q + n \cdot B^\top B + Q^\top \vec{1}\vec{1}^\top B + B^\top \vec{1}\vec{1}^\top Q + \lambda I \right)^{-1}$$
$$\left(\sum_i^n \sum_c^l W_{u,i,c} X_{u,i,c} (Q_i + B_c)^\top + \sum_i^n \sum_c^l W'_{u,i,c} Q_i B_c^\top \cdot (Q_i + B_c)^\top + Q^\top Q B^\top \vec{1} + B^\top B Q^\top \vec{1} \right)$$

The computational complexity is equal to that of iTALS: $O(k^3(m+n+l) + k^2 p)$. Notice that there is no 'one' variant for iTALSx, because in this decomposition, multiplication with zero vectors is actually the identity operation. I.e. if all context factors are zero, the prediction is solely based on the user-item part. Furthermore, we were unable to replicate the multidimensional extension of PITF for CARS, FTF [5], or any of its variants with reasonable effort due to missing source code and its complex model formulation. Additionally, the method scales linearly with tensor size which is not desirable.

2.3 TTF Decomposition: WTF

Our novel contribution, Weighted Tensor Factorization (WTF), is based on the Tensor Train Factorization [18]. Its structure is similar to iTALSs with the exception of context factors being full matrices instead of only vectors (or diagonals):

$$\mathcal{L}^{(WTF)} = \sum_u^m \sum_i^n \sum_c^l W_{u,i,c} \left(X_{u,i,c} - P_u B^{(c)} Q_i^\top \right)^2 + \lambda \left(\|P\|_F^2 + \|Q\|_F^2 + \sum_c^l \left\| B^{(c)} - I \right\|_F^2 \right)$$

$$P_u = \left(\sum_c^l B^{(c)} Q^\top Q B^{(c)^\top} + \sum_i^n \sum_c^l W'_{u,i,c} B^{(c)} Q_i^\top Q_i B^{(c)^\top} + \lambda I \right)^{-1} \left(\sum_i^n \sum_c^l W_{u,i,c} X_{u,i,c} B^{(c)} Q_i^\top \right)$$

The update equation for item factors is similar, but with $B^{(c)}$ transposed. For the context matrices $B^{(c)}$ we find:

$$P^\top P B^{(c)} Q^\top Q + \sum_u^m \sum_i^n W'_{u,i,c} P_u^\top P_u B^{(c)} Q_i^\top Q_i + \lambda B^{(c)} = \sum_u^m \sum_i^n W_{u,i,c} X_{u,i,c} P_u^\top Q_i + \lambda I$$

This can either be solved by vectorization [16] or with approximate sparse solvers, such as the conjugate gradient method (CG) [9]. The computational complexity for an update step of the user and item factors is $O(k^3(m+n)+k^2 p)$, the same as for iTALS(x). Updating all the context matrices with an exact method (vectorization) is more expensive: $O(k^6 c + k^2 p)$. However, it is found that an approximation of the update step often suffices for the algorithm to converge [25]. This is not surprising, as the computed value is overwritten in each iteration anyway, so computing it with high precision is not needed. Computing the context matrices with the CG method instead of vectorization reduces the complexity back to cubic in practice because only a small amount of CG steps (typically two or three [25]) are needed for adequate convergence. Note that approximate solvers can also be leveraged in the other methods to make them more efficient. For WTF however, their use is *required* to make the method computationally feasible for reasonably sized datasets. There is again a 'one' variant of WTF with regularization of the context factors to the identity matrix, similar to iTALS one.

2.4 Frequency-based Regularization

To keep the notation simple, we depicted a single scalar parameter λ for regularizing all factors. In practice, it is often found that scaling the strength of the regularization with the amount of observations improves performance [22]. Hence, we adapted the hyperparameter $\nu \in [0, 1]$ of [22], which is an exponent over the sum of weights associated with the factor in the loss. With this parameter and the global λ scalar we compute the regularization strength per factor. For example for the user factor P_u in the iTALS model the regularization becomes: $\lambda \cdot \left(\sum_i^n \sum_{c_1}^{l_1} \cdots \sum_{c_d}^{l_d} W_{u,i,c_1,\ldots,c_d} \right)^\nu \|P_u\|_2^2$.

Table 2. Statistics of the datasets after preprocessing. Context features indicates the amount of unique values per variable.

Dataset	Users	Items	Interactions	Context features
Frappe	816	4 058	96 002	7 + 7 + 7
TripAdvisor	2 362	2 221	13 258	79 + 5
Food.com	22 178	15 086	388 362	4 + 7

3 EXPERIMENTS

3.1 Datasets

The different models are compared on three publicly available datasets. We restrict the scope of this study to purely contextual attributes that 1) depend on the interaction and not only on the user or only on the item and 2) are known to the system at the time of recommendation. In other words, we do not use item or user metadata for context or preferences that cannot easily be derived such as the time a user has available for cooking. It should be noted that, compared to popular datasets for evaluating non-context aware recommendation, unfortunately the publicly available datasets for this task are rather small [15]. The datasets are described below and their statistics are summarized in Table 2.

Frappe: Commonly used dataset in CARS research [3, 15]. It contains app usage logs that record *time of day*, *weekday* and the *weather*. Though it does not have many users, the context features are very descriptive.

TripAdvisor: A hotel rating dataset commonly used in CARS [2, 15]. For context we use the *state* (or country) the user is in and the *trip type*.

Food.com: A rating and review dataset of recipes [17]. Here, we manually extract context features based on the date attribute, namely *season* and *weekday*.

Rating datasets were made binary by choosing ratings of 3 out of 5 and above as positive. For all datasets we only retain users with at least 3 items and for the Food.com dataset we also dropped items with less than 10 interactions. Lastly, none of the datasets contain missing values for contexts, except for Frappe where the *weather* attribute has 13% missing values.

3.2 Experimental Setup

For our evaluation, we use disjunct training and testing interactions, but not disjunct train and test users. In this setup, the interactions of each user are assigned to one of these two sets. To keep the evaluation simple and fair, we chose the leave-one-out strategy where one interaction with an item is assigned to the test set and all others to the training set. With this approach, our prediction step becomes easier as only one top-k list needs to be computed per user-context pair of the left-out interaction, and hence all users contribute equally to the computed metrics. This evaluation scenario is both a good choice for simulating the use of factorization models, where user factors need to be learned beforehand, and also to make the most of the limited amount of data available. Additionally, retargetting already consumed items is prevented except for Frappe (app usage contains repeats). First, hyperparameters are optimized using grid search on a single split. Then, a 5-fold cross validation on the entire dataset is performed to compute the final metrics and their standard deviation.

Two standard metrics in information retrieval are used to evaluate the models: Hit Rate (HR@k) and Mean Reciprocal Rank (MRR@k). HR@k computes the ratio of true positives that are included in the top-k recommendation list (without taking rank into account). Since we only have one left-out interaction this is either one or zero per user depending on whether the left-out item was recommended. Taking the average over all users gives the percentage of users that got a relevant recommendation. MRR@k is a ranking metric that takes the rank of the first hit (true positive) in the

top-k recommendation list into account by weighing its contribution with one over the rank. Its value is hence lower than or equal to the HR in our setup. As top-k list size we chose 5 and 20 for both metrics which are typical choices for evaluating recommender systems. Alongside our CARS models, we also compare with three context-unaware baselines:
ItemKNN [6]: Item-based similarity model with cosine similarity.
EASE [24]: State-of-the-art autoencoder based recommender with closed-form solution.
WMF [14, 20, 22]: State-of-the-art matrix factorization for implicit feedback data.

3.3 Results

First, we compare the baselines and all CARS methods on the three datasets as shown in Table 3a. Ten training steps and $k = 80$ for the latent dimension are used which achieved convergence and optimal results for these relatively small datasets. We observe that Frappe shows the biggest increase in performance when using context features. For TripAdvisor a small but consistent improvement can be observed, and finally, for the Food.com dataset most models perform worse than the best baseline except for two that perform about equally well. From this, it is clear that the context of the Food.com dataset is less informative for computing recommendations. Despite these non-informative or misleading context features that appear to not generalize well, it is still interesting to see that WTF one and iTALS one can match the best baseline. Especially since the performance of all other models decreases by taking context into account. We conclude that WTF one and iTALS one are the most robust to irrelevant information. This also makes sense intuitively as the regularization of these models pushes them towards learning offsets from the standard user-item prediction. Indeed, if there is no clear signal in the context features, their factors will not deviate from the prediction based on only the user and item.

Second, we find that the 'one' variants of the models outperform their respective classically regularized variants on all datasets. A possible explanation for this is that the improved regularization is more appropriate for the modelling task and in general a better match for the role contextual data plays in recommendation. Where we expect a strong influence from users to items as both entities have unique characteristics and preferences, this is simply not the case for most context features. For example, if a user wants to use a specific app, then whether it is rainy or sunny outside or what the current time is will likely only slightly influence this preference. Modelling the context features as offsets on an already established user-item preference (rather than an equally important entity) makes more sense from this point of view.

An extreme case of this phenomenon can be observed when comparing iTALS with iTALS one. It is clear that iTALS is not able to model any of the three datasets, likely due to the compounding effect of treating context combinations as equally important entities. First, by modelling combinations of contexts, the data to learn from becomes even more sparse as the dimensions increase but the amount of positive interactions remains the same. Second, because context features are modelled the same as users and items in the loss, they are given the same importance, or even more importance as the number of context dimensions increases. Except for specific situations such as user cold-start, this is likely not desirable for the recommendation task. iTALS one prevents this behaviour by encouraging the context factors to only model offsets to the user and item predictions.

For the Frappe dataset specifically, the poor performance of iTALS can also in part be explained by the missing features. Since factors of missing features are set to the zero vector, the entire score for all items will be zero and no sensible ranking can be made. iTALS one and iTALSs do not have this problem because the missing context factor is set to one in the former and the average score is taken with other contexts in the latter.

Third, to investigate flattening context features over modelling them multidimensionally we can compare iTALSs one with iTALS one. On the Frappe and TripAdvisor datasets we find no improvement to modelling combinations of contexts. Again this can mainly be attributed to the data becoming exponentially more sparse as the number of

Table 3. Experimental results on three datasets with best result(s) marked in bold. The left and right tables are different experiments.

(a) Context-unaware (Base), 3D CARS (Flat), and multidimensional CARS (MD). Standard deviation rounded to thousandths given between brackets.

(b) CARS with context-unaware user-item factors. Percentage of original performance marked inside brackets.

	Frappe	MRR@5	MRR@20	HR@5	HR@20	MRR@5	MRR@20	HR@5	HR@20
Base	ItemKNN	.071 (.006)	.101 (.009)	.155 (.007)	.451 (.020)				
	EASE	.179 (.006)	.204 (.007)	.345 (.011)	.584 (.014)				
	WMF	.198 (.010)	.224 (.010)	.368 (.010)	.619 (.011)	.198 (100%)	.224 (100%)	.368 (100%)	.619 (100%)
Flat	iTALSs	.276 (.005)	.296 (.004)	.456 (.006)	.638 (.008)	.198 (72%)	.226 (76%)	.362 (79%)	.626 (98%)
	iTALSs one	**.287** (.007)	**.308** (.006)	.445 (.014)	.634 (.011)	.199 (69%)	.226 (73%)	.365 (82%)	.627 (99%)
	iTALSx	.283 (.006)	.302 (.005)	**.467** (.009)	.639 (.007)	.204 (72%)	.230 (76%)	.376 (81%)	.625 (98%)
	WTF	.264 (.008)	.283 (.007)	.416 (.015)	.591 (.016)	.240 (91%)	.265 (94%)	**.417** (100%)	**.650** (110%)
	WTF one	.274 (.005)	.296 (.005)	.450 (.007)	**.656** (.009)	**.242** (88%)	**.266** (90%)	.416 (92%)	.647 (99%)
MD	iTALS	.079 (.004)	.096 (.003)	.155 (.006)	.330 (.005)	.165 (209%)	.187 (195%)	.299 (193%)	.515 (156%)
	iTALS one	.280 (.009)	.301 (.007)	.461 (.015)	.652 (.010)	.198 (71%)	.225 (75%)	.367 (80%)	.627 (96%)

	TripAdvisor	MRR@5	MRR@20	HR@5	HR@20	MRR@5	MRR@20	HR@5	HR@20
Base	ItemKNN	.006 (.001)	.008 (.001)	.011 (.001)	.040 (.002)				
	EASE	.020 (.001)	.025 (.001)	.037 (.002)	.085 (.004)				
	WMF	.022 (.002)	.028 (.002)	.040 (.004)	.102 (.002)	.022 (100%)	.028 (100%)	.040 (100%)	.102 (100%)
Flat	iTALSs	.009 (.001)	.012 (.001)	.018 (.002)	.056 (.003)	.023 (256%)	.029 (242%)	.044 (244%)	.104 (186%)
	iTALSs one	.023 (.002)	.029 (.001)	.041 (.003)	.104 (.001)	.023 (100%)	.028 (97%)	.043 (105%)	.105 (101%)
	iTALSx	**.026** (.002)	**.032** (.002)	**.047** (.004)	**.110** (.007)	.024 (92%)	.029 (91%)	.044 (94%)	**.106** (96%)
	WTF	.022 (.002)	.027 (.001)	.039 (.002)	.094 (.003)	.024 (109%)	.029 (107%)	.043 (110%)	.102 (109%)
	WTF one	.025 (.002)	.031 (.002)	.045 (.004)	**.110** (.002)	**.025** (100%)	**.030** (97%)	**.045** (100%)	**.106** (96%)
MD	iTALS	.006 (.002)	.009 (.002)	.014 (.002)	.043 (.003)	.023 (383%)	.029 (322%)	.044 (314%)	.104 (242%)
	iTALS one	.021 (.002)	.026 (.002)	.039 (.004)	.096 (.002)	.023 (110%)	.028 (108%)	.043 (110%)	.105 (109%)

	Food.com	MRR@5	MRR@20	HR@5	HR@20	MRR@5	MRR@20	HR@5	HR@20
Base	ItemKNN	.010 (.001)	.012 (.001)	.016 (.001)	.030 (.001)				
	EASE	**.016** (.001)	.018 (.001)	.026 (.001)	.056 (.001)				
	WMF	.015 (.000)	**.019** (.000)	.026 (.001)	.063 (.001)	**.015** (100%)	**.019** (100%)	**.026** (100%)	**.063** (100%)
Flat	iTALSs	.011 (.000)	.014 (.000)	.021 (.001)	.053 (.001)	.013 (118%)	.017 (121%)	.023 (110%)	.057 (108%)
	iTALSs one	.014 (.000)	.017 (.000)	.024 (.001)	.058 (.002)	.013 (93%)	.017 (100%)	.023 (96%)	.057 (98%)
	iTALSx	.014 (.000)	.017 (.000)	.024 (.001)	.059 (.001)	.014 (100%)	.017 (100%)	.024 (100%)	.058 (98%)
	WTF	.011 (.000)	.014 (.000)	.020 (.001)	.047 (.001)	.014 (127%)	.018 (129%)	.025 (125%)	.060 (128%)
	WTF one	**.016** (.000)	**.019** (.000)	**.028** (.001)	**.065** (.001)	.014 (88%)	.017 (89%)	.024 (86%)	.057 (88%)
MD	iTALS	.007 (.001)	.009 (.001)	.014 (.001)	.035 (.007)	.013 (186%)	.016 (178%)	.023 (164%)	.054 (154%)
	iTALS one	**.016** (.000)	**.019** (.000)	**.028** (.000)	**.065** (.000)	.013 (81%)	.017 (89%)	.023 (82%)	.057 (88%)

dimensions grows. In most cases it also seems more intuitive to model contexts as independent because we do not expect the information to drastically change when contexts occur together. For example, it may be raining in the evening or raining in the morning but using the offset of raining plus the offset of the time will likely be a good proxy for both cases. If dependencies were present in the data (and if there would be enough data to learn them from), then we can expect multi-dimensional models to outperform their flattened variants.

Finally, the iTALSx model achieves good results across all datasets. Only on the Food.com dataset we can see it is outperformed by WTF one and iTALS one, indicating these models may have a higher modelling capacity or could be less prone to noise. More data is needed to support these hypotheses.

3.4 Results with Context-unaware User-item Factors

In a second experiment we investigate whether context factors can be learned post-hoc with pre-learned user and item factors. Given that many recommender systems already use embeddings, if the context factors can be learned after the training procedure that is already in place, they are easier to integrate into the system. Additionally, the training cost is reduced as the context factors only need one (or very few for multi-dimensional) iteration(s) and the learning of user and item factors does not need to take context into account.

The results of this experiment are reported in Table 3b where we used the optimal factors of WMF for each dataset followed by only learning the context factors of each method in one iteration (three for iTALS and iTALS one to learn dependencies between contexts). Hyperparameters were optimized again for learning the contexts. Though this training procedure is significantly faster, it is clear from the results that not all context information can be extracted in this setup. The methods that perform better than when they were trained with context from the start are the ones that got results worse than the WMF baseline and were now able to achieve similar or slightly higher metrics than the context-unaware baseline. However none of the methods can match the optimal results of the previous experiment as can very clearly be seen for the Frappe dataset and also for TripAdvisor to a lesser extent. For the Food.com dataset there is even less information to be learned from contexts and all results are close to the WMF baseline.

If we look at which method is best suited for post-hoc learning we find that WTF and WTF one are able to make the best of this situation. This is not surprising given the fact their context factors contain more weights and are able to model more complex ‘transformations’ between the user and item factors. Nevertheless, there remains a benefit to learning user and item factors with contexts in the loss because the context under which a user consumed an item can influence those factors in a significant way. For example, if two users consumed the same items but under different contexts, their factors will converge to the same optimum and even the best and most descriptive context factor cannot separate them post-hoc.

4 CONCLUSIONS

Tensor decompositions prove to be a powerful and intuitive tool for context-aware recommendation. Their linear scaling with the number of users, items and contexts, and easy parallellization allow them to remain computationally efficient while learning from the entire dataset without need for negative-sampling. In this paper, first we introduce a new regularization strategy that is more appropriate for the recommendation task. Second, we derive a new model based on the TTF decomposition. Third, we compare different decomposition methods, tensor structures and regularization strategies for CARS and demonstrate the effect those three choices have depending on the use case. If many context features are available that potentially depend on each other, then a multidimensional model with ‘one’ regularization likely works best. Otherwise, if there are few context variables that are very rich, a model like ‘WTF one’ that learns powerful context factors is more applicable. Understanding the context and requirements of the recommender system is key to supplying good recommendations, as there is no single method that excels at every use case. This paper serves as a practical guide for the trade-offs to consider with weighted tensor decompositions for CARS.

ACKNOWLEDGMENTS

This work was supported by the Research Foundation — Flanders (FWO) [11E5921N to J. De Pauw].

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2010. Context-aware recommender systems. In *Recommender systems handbook*. Springer, 217–253.

[2] Md Hijbul Alam, Woo-Jong Ryu, and SangKeun Lee. 2016. Joint multi-grain topic sentiment: modeling semantic aspects for online reviews. *Information Sciences* 339 (2016), 206–223.

[3] Linas Baltrunas, Karen Church, Alexandros Karatzoglou, and Nuria Oliver. 2015. Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild. *arXiv preprint arXiv:1505.03014* (2015).

[4] Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu, and Shaoping Ma. 2020. Efficient neural matrix factorization without sampling for recommendation. *ACM Transactions on Information Systems (TOIS)* 38, 2 (2020), 1–28.

[5] Szu-Yu Chou, Jyh-Shing Roger Jang, and Yi-Hsuan Yang. 2018. Fast tensor factorization for large-scale context-aware recommendation from implicit feedback. *IEEE Transactions on Big Data* 6, 1 (2018), 201–208.

[6] Mukund Deshpande and George Karypis. 2004. Item-Based Top-N Recommendation Algorithms. *ACM Trans. Inf. Syst.* 22, 1 (Jan. 2004), 143–177. https://doi.org/10.1145/963770.963776

[7] Evgeny Frolov and Ivan Oseledets. 2017. Tensor methods and recommender systems. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 7, 3 (2017), e1201.

[8] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 549–558.

[9] Magnus R Hestenes and Eduard Stiefel. 1952. Methods of conjugate gradients for solving. *Journal of research of the National Bureau of Standards* 49, 6 (1952), 409.

[10] Balázs Hidasi. 2014. Factorization models for context-aware recommendations. *Infocommun J VI (4)* (2014), 27–34.

[11] Balázs Hidasi and Domonkos Tikk. 2012. Fast ALS-based tensor factorization for context-aware recommendation from implicit feedback. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 67–82.

[12] Balázs Hidasi and Domonkos Tikk. 2016. General factorization framework for context-aware recommendations. *Data Mining and Knowledge Discovery* 30, 2 (2016), 342–371.

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

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

[15] Sergio Ilarri, Raquel Trillo-Lado, and Ramón Hermoso. 2018. Datasets for context-aware recommender systems: Current context and possible directions. In *2018 IEEE 34th International Conference on Data Engineering Workshops (ICDEW)*. IEEE, 25–28.

[16] Peter Lancaster and Miron Tismenetsky. 1985. *The theory of matrices: with applications*. Elsevier.

[17] Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley. 2019. Generating personalized recipes from historical user preferences. *arXiv preprint arXiv:1909.00105* (2019).

[18] Ivan V Oseledets. 2011. Tensor-train decomposition. *SIAM Journal on Scientific Computing* 33, 5 (2011), 2295–2317.

[19] Roberto Pagano, Paolo Cremonesi, Martha Larson, Balázs Hidasi, Domonkos Tikk, Alexandros Karatzoglou, and Massimo Quadrana. 2016. The contextual turn: From context-aware to context-driven recommender systems. In *Proceedings of the 10th ACM conference on recommender systems*. 249–252.

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

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

[22] Steffen Rendle, Walid Krichene, Li Zhang, and Yehuda Koren. 2022. Revisiting the performance of ials on item recommendation benchmarks. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 427–435.

[23] Steffen Rendle and Lars Schmidt-Thieme. 2010. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*. 81–90.

[24] Harald Steck. 2019. Embarrassingly shallow autoencoders for sparse data. In *The World Wide Web Conference*. 3251–3257.

[25] Gábor Takács, István Pilászy, and Domonkos Tikk. 2011. Applications of the conjugate gradient method for implicit feedback collaborative filtering. In *Proceedings of the fifth ACM conference on Recommender systems*. 297–300.

[26] Ledyard R Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika* 31, 3 (1966), 279–311.