Multiverse Recommendation: N-dimensional Tensor Factorization for Context-aware Collaborative Filtering

Alexandros Karatzoglou
Telefonica Research
Barcelona, Spain
alexk@tid.es

Xavier Amatriain
Telefonica Research
Barcelona, Spain
xar@tid.es

Linas Baltrunas
Free University of Bolzano
Bolzano, Italy
lbaltrunas@unibz.it

Nuria Oliver
Telefonica Research
Barcelona, Spain
nuriao@tid.es

ABSTRACT

Context has been recognized as an important factor to consider in personalized Recommender Systems. However, most model-based Collaborative Filtering approaches such as Matrix Factorization do not provide a straightforward way of integrating context information into the model. In this work, we introduce a Collaborative Filtering method based on *Tensor Factorization*, a generalization of Matrix Factorization that allows for a flexible and generic integration of contextual information by modeling the data as a User-Item-Context N-dimensional tensor instead of the traditional 2D User-Item matrix. In the proposed model, called *Multiverse Recommendation*, different types of context are considered as additional dimensions in the representation of the data as a tensor. The factorization of this tensor leads to a compact model of the data which can be used to provide context-aware recommendations.

We provide an algorithm to address the N-dimensional factorization, and show that the *Multiverse Recommendation* improves upon non-contextual Matrix Factorization up to 30% in terms of the Mean Absolute Error (MAE). We also compare to two state-of-the-art context-aware methods and show that Tensor Factorization consistently outperforms them both in semi-synthetic and real-world data – improvements range from 2.5% to more than 12% depending on the data. Noticeably, our approach outperforms other methods by a wider margin whenever more contextual information is available.

Categories and Subject Descriptors

H3.3 [**Information Search and Retrieval**]: Information filtering—*Collaborative Filtering*; H3.4 [**Systems and Software**]: Performance evaluation (efficiency and effectiveness); G3 [**Probability and Statistics**]: Correlation and regression analysis; G1.6 [**Optimization**]: Gradient methods

General Terms

Algorithms, Experimentation

Keywords

Collaborative Filtering, Tensor Factorization, Context

1. INTRODUCTION

Recommender Systems have become ubiquitous tools to find our way in the age of information. User preferences are inferred from past consumption patterns or explicit feedback and predictions are computed by analyzing other users – *Collaborative Filtering (CF)* – or categorizing the items by their content – *Content-based Recommendations*. The simplest formulation of the *recommendation problem* involves predicting values for $user, item$ pairs. In the CF setting, this turns the problem into a sparse two-dimensional matrix in which we have very few values for some $user, item$ pairs that must be used to compute the missing values of interest.

The two main CF approaches that emerged as victorious from the Netflix Prize were *neighborhood methods* and *latent factor models*. Neighborhood methods use similarity functions such as the *Pearson Correlation* or *Cosine Distance* to compute sets of neighbors to a user or an item. Recommendations are then computed by using data from those neighbors. On the other hand, latent factor models [5] such as *Matrix Factorization* (MF) solve the recommendation problem by decomposing the $user, item$ matrix and learning latent factors for each user and item in the data. The underlying assumption is that both users and items can be modeled by a reduced number of factors. This approach has proven to be the most accurate method in isolation in different settings.

Although the simplified $user, item$ recommendation model can be used successfully in many settings, it is not uncommon to find real settings in which additional variables come into play. For instance, there are many situations where *time* plays an important role in defining a user's preference for an item. In this case, the two- dimensional matrix is turned into a three dimensional $user, item, time$ tensor. The set of

RecSys2010, September 26–30, 2010, Barcelona, Spain.

variables that influence the user's preference for a given item are referred to as *context.*

In this paper, we present a generic CF model that is based on a generalization of MF to address contextual recommendation problems. To this end, we extend the concept of matrix factorization to that of *tensor factorization* (see section 3.1). A tensor is a generalization of the matrix concept to multiple dimensions. In the example given above, the usual $user, item$ two-dimensional matrix is converted into a three-dimensional tensor (see Figure 1). Tensor Factorization (TF) can be used to add any number of variables to a recommender system. In particular, it could be used to hybridize content and CF in a way similar to the approach by Pilaszy and Tikk [16]. However, in this work we focus on the use of TF for adding contextual information. We call our approach to contextual recommendation via TF *Multiverse Recommendation* because it can be used to efficiently bridge hidden "worlds" separated by different contexts and therefore collapse parallel dimensions into a coherent model.

The relation of our model to other latent models and context-aware systems is reviewed in section 2, and our experimental results are summarized in section 4, where we validate the proposed approach on several synthetic and real world datasets. In addition, we empirically verify that our approach improves the baseline when not taking into account the context and show how our method outperforms state-of-the-art contextual approaches.

The proposed model brings a number of contributions to the area of contextual recommendations, including the ability to:

- Generalize efficient MF approaches to the N-dimensional case in a compact way
- Include any number of contextual dimensions into the model itself
- Benefit from several loss functions designed to finetune the optimization criteria
- Train the model with a fast and straightforward algorithm
- Take advantage of the sparsity of the data while still exploiting the interaction between all users-items and context.

2. RELATED WORK

Although most of the work on Collaborative Filtering has focused on the traditional two-dimensional user/item problem, there has been a recent increase of interest in adding context to recommendations. This is probably due to the relevance of context in mobile applications and the success that some applications have had using some contextual variable such as location (*e.g.* Foursquare [1]).

Adomavicius and Tuzhilin [2] do a thorough review of approaches to contextual recommendations in their book chapter and categorize context-aware recommender systems (CARS) into three types: *contextual pre-filtering*, where context drives data selection; *contextual post-filtering*, where context is used to filter recommendations once they have been computed using a traditional approach; and *contextual modeling*, where context is integrated directly into the model. An example of contextual pre-filtering is the so-called *user micro-profile*, in which a single user is represented by a hierarchy of possibly overlapping contextual profiles [3]. Post-filtering methods can use traditional approaches and then apply filtering or weighting. In their experimental evaluation, Panniello *et al.* [15] found that the choice of a pre-filtering or post-filtering strategy depends on the particular method and sometimes a simple post-filter can outperform an elaborate pre-filtering approach.

The approach proposed in this paper belongs to the last category of contextual modeling. Although some standard model-based approaches could theoretically accept more dimensions, the only models to report results in this category are Oku *et al.*'s Context-aware Support Vector Machines (SVM) [12]. The authors consider support vectors in a multidimensional space and find the separating hyperplane. Their experiments show that contextual recommendations perform better than non-contextual. Other authors that have addressed the issue of context-aware recommendations from a multidimensional perspective are Adomavicius *et al.* [1], who introduce a multidimensional model based on the OnLine Analytical Processing technique for decision support which still is based on a pre-filtering method; and Palmisano *et al.* [14] who describe contextual information with K dimensions, each of them having q attributes.

Factorization models have recently become one of the preferred approaches to Collaborative Filtering, especially when combined with neighborhood methods [8]. However, even when processing datasets such as the Netflix Prize, the importance of context has become clear. Koren has recently introduced a temporal model into the factor model called *timeSVD++* [9] which significantly improves the Root Mean Squared Error (RMSE) on the Netflix data compared to previous non-temporal factorization models.

Standard TF, including our proposed approach, have the assumption of variables over finite and categorical domains. Xiong *et al.* present a Bayesian Probabilistic TF wich tries to overcome this limitation. The authors model the temporal evolution of online shopping preferences [22]. Their experiments show that using time as the third dimension in the form of a tensor does improve accuracy when compared to the non-temporal case. As we will explain later, Xiong's model can in fact be considered as a simple case instance of the more generic model presented in this paper. Finally, three dimensional TF has also been proposed by Rendle *et al.*, for Collaborative tag Recommendation. Rendle *et al.* use the third tensor dimension in their $user, item, tag$ model to represent the target variable which in their case are tags coded as binary vectors where the presences of a tag is market by a 1 and the absence by a 0 [17].

However and to the best of our knowledge, there is no previous work on the use of N-dimensional tensors for CF, which is the main contribution of this paper and will be presented next in detail.

3. MULTIVERSE RECOMMENDATION

TF is an existing N-dimensional extension of Matrix Factorization. However, a straightforward use of this model makes it unsuitable for the CF case. In the current section we introduce the model of Matrix and Tensor Factorization and explain the details of how we have adapted this model for N-dimensional CF.

The main idea behind the use of TF is that we can take ad-

[1] http://www.foursquare.com

vantage of the same principles behind Matrix Factorization to deal with N-dimensional information. However, before we dive into the details of the TF model, we shall briefly summarize the two-dimensional MF approach.

Matrix Factorization.

CF techniques based on MF work by assuming that ratings provided by users on items can be represented in a sparse matrix $Y \in \mathcal{Y}^{n \times m}$ where (n is the number of users and m the number of items). The observed values in Y are thus formed by the rating information provided by the users on the items. The CF problem then boils down to Matrix Completion problem. In MF techniques the aim is to factorize the matrix of observed ratings into two matrices $U \in \mathbb{R}^{n \times d}$ and $M \in \mathbb{R}^{m \times d}$ such that $F := UM^\top$ approximates Y, *i.e.* minimizes a loss function $L(F, Y)$ between observed and predicted values. In most cases, a regularization term for better generalization performance is added to the loss function and thus the objective function becomes $L(F, Y) + \Omega(F)$. Standard choices for L include the least squares loss function $L(F, Y) = \frac{1}{2}(F - Y)^2$ and for Ω the Frobenius norm *i.e.* $\Omega(F) = \lambda \left[\|U\|^2_{\text{Frob}} + \|M\|^2_{\text{Frob}}\right]$.

3.1 Tensor Factorization

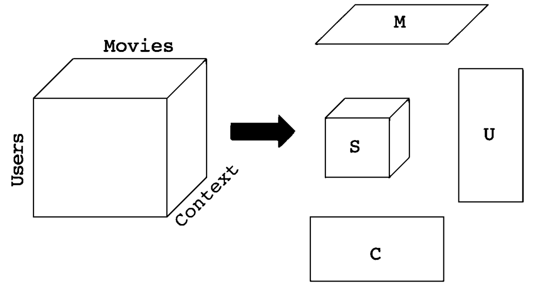


Figure 1: Illustration of the (3-dimensional) HOSVD tensor factorization model.

N-dimensional TF is a generic model framework for recommendations that is able to handle N-dimensional data and profit from most of the advantages of MF, such as fast prediction computations and simple and efficient optimization techniques. We shall now introduce the generic TF model and describe its specific adaptation to context-aware CF in Section 3.2.

Notation.

For the sake of simplicity, we will describe the model for a single contextual variable C, and therefore Y the tensor containing the ratings will be a 3-dimensional tensor. The generalization to larger numbers of context variables and N dimensions is trivial. In the following we denote the sparse tensor of rating observations by $Y \in \mathcal{Y}^{n \times m \times c}$, where n are the number of users, m the number of items, and c where $c_i \in \{1, \dots, c\}$ the number of contextual variables. Typically, the rating is given on a five star scale and thus $Y \in \{0, \dots, 5\}^{n \times m \times c}$, where the value 0 indicates that a user did not rate an item. In this sense, 0 is special since it does *not* indicate that a user dislikes an item but rather that data is missing. We are using two tensor operations a tensor-matrix multiplication operator denoted by $\times_U$ where the subscript shows the direction on the tensor on which to multiply the matrix i.e $T = Y \times_U U$ is $T_{ljk} = \sum_{i=1}^{n} Y_{ijk} U_{ij}$ and the tensor outer product denoted by $\otimes$. Finally, $D \in \{0; 1\}^{n \times m \times c}$ is a binary tensor with nonzero entries D_{ijk} whenever Y_{ijk} is observed and we denote by U_{i*} the entries of the i_{th} row of matrix U.

HOSVD-decomposition.

There are different types of tensor decomposition models in the literature, such as the Canonical Decomposition or Parallel Factors – which is also known as the CP-decomposition – and the High Order Singular Value decomposition (HOSVD). In our approach, we follow the HOSVD formulation shown in Figure 1, where the 3-dimensional tensor is factorized into three matrices $U \in \mathbb{R}^{n \times d_U}$, $M \in \mathbb{R}^{m \times d_M}$ and $C \in \mathbb{R}^{c \times d_C}$ and one central tensor $S \in \mathbb{R}^{d_U \times d_M \times d_C}$. In this case, the decision function for a single *user i, item j, context k* combination becomes:

$$F_{ijk} = S \times_U U_{i*} \times_M M_{j*} \times_C C_{k*} \tag{1}$$

This decomposition model allows for full control over the dimensionality of the factors extracted for the users, items and context by adjusting the d_u, d_M and d_C parameters. This property is valuable in the case of large real world datasets where the matrices U and M can grow in size and potentially pose a storage problem.

3.2 Tensor Factorization for CF

The aim in proposing an N-dimensional TF approach for context-based recommendation is to model the context variables in the same way as the users and items are modeled in MF techniques by taking the interactions between users-items-context into account. We shall refer to this approach as *Multiverse Recommendations.*

Existing HOSVD methods (e.g. [10]) require a dense matrix Y and therefore ignore the sparsity of the input data. Treating Y as a dense tensor with missing entries being assumed to be 0, would introduce a bias against unobserved ratings. The model of Regularized TF introduced in this section avoids these issues by optimizing only the observed values in the rating tensor. Also note that, in contrast to standard SVD and HOSVD methods, in CF there is no need for imposing orthogonality constrains on the factors.

Multiverse Recommendations have a number of advantages compared to many of the current context-based methods, including: (1) *No need for pre- or post-filtering*: In contrast to many of the current algorithms which rely on splitting and pre- or post-filtering the data based on context, TF utilizes all the available ratings to model the users and the items. Splitting or pre-or post- filtering the data based on the context can lead to loss of information about the interactions between the different context settings; (2) *Computational simplicity:* Many of the proposed methods rely on a sequence of techniques which often prove to be computationally expensive rather than on a single and less computationally expensive model, as is the case in TF; (3) *Ability to handle N-dimensions:* Moreover, the TF approach we introduce generalizes well to an arbitrary amount of context variables while adding relatively little computational overhead.

3.2.1 Loss Function

In analogy to MF approaches [18, 21], we define the loss function as:

$$L(F,Y) := \frac{1}{\|S\|_1} \sum_{i,j,k} D_{ijk} l(F_{ijk}, Y_{ijk}) \tag{2}$$

where $l : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$ is a pointwise loss function penalizing the distance between estimate and observation and F_{ijk} is given by equation 1. Note that the total loss L is defined only on the observed values in the tensor Y. Possible choices for the loss function l include the following:

Squared error: Provides an estimate of the conditional mean

$$l(f,y) = \frac{1}{2}(f-y)^2 \text{ and}$$
$$\partial_f l(f,y) = f - y$$

Absolute loss: Provides an estimate of the conditional median

$$l(f,y) = |f-y| \text{ and}$$
$$\partial_f l(f,y) = \operatorname{sgn}[f-y]$$

ϵ-insensitive loss: Chosen to ignore deviations of up to ϵ via

$$l(f,y) = \max(0, |y-f| - \epsilon) \text{ and}$$
$$\partial_f l(f,y) = \begin{cases} \operatorname{sgn}[f-y] & \text{if } |f-y| > \epsilon \\ 0 & \text{otherwise} \end{cases}$$

These are not the only loss functions possible. For instance, by using a quantile loss we can prioritize estimates based on the confidence with which we would like to recommend events. Other possible loss functions include the Huber loss [7] and the hinge loss function, which can be useful in the case of implicit taste information [20].

3.2.2 Regularization

Simply minimizing a loss function is known to lead to overfitting. Given the factors U, M, C, S which constitute our model, we have a choice of ways to ensure that the model complexity does not grow without bound. A simple option is to add a regularization term based on the l_2 norm of these factors [19]. In the case of a matrix, this norm is also known as the Frobenius norm.

$$\Omega[U,M,C] := \frac{1}{2}\left[\lambda_U \|U\|^2_{\text{Frob}} + \lambda_M \|M\|^2_{\text{Frob}} + \lambda_C \|C\|^2_{\text{Frob}}\right]. \tag{3}$$

In a similar manner, we can also restrict the complexity of the central tensor S by imposing a l_2 norm penalty:

$$\Omega[S] := \frac{1}{2}\left[\lambda_S \|S\|^2_{\text{Frob}}\right]. \tag{4}$$

Note here that one can also impose an ℓ_1 norm as a regularizer which is known to lead to sparse solutions [11, 6]:

$$\Omega[U,M,C] = \sum_{id_U} |U_{id_U}| + \sum_{id_M} |M_{id_M}| + \sum_{id_C} |C_{cd_C}| \tag{5}$$

Even though regularizer in Eq.5 leads to particularly sparse models, optimization is non-trivial. During the course of the optimization process, a potentially large number of parameters is needed to make progress. Hence, in this work we use regularizers as in eq. 3 for U, M, C and eq. 4 for S.

3.2.3 Optimization

Overall, we strive to minimize a regularized risk functional that is a combination of $L(F,Y)$ and $\Omega[U,M,C]$. The objective function for the minimization problem is:

$$R[U,M,C,S] := L(F,Y) + \Omega[U,M,C] + \Omega[S] \tag{6}$$

Minimizing this objective function can be done using many approaches. Subspace descent is a popular choice in MF methods and could be used in the tensor setting. In subspace descent one optimizes iteratively over individual components of the model while keeping the remaining components fixed, *e.g.* optimize over the U matrix while keeping the remaining matrices and tensor fixed, then over M etc. This method leads to quick convergence but requires the optimization procedure to be run in a batch setting.

As dataset sizes grow, it becomes increasingly infeasible to solve factorization problems by batch optimization. Instead, we resort to a simple online algorithm which performs stochastic gradient descent (SGD) in the factors U_{i*}, M_{j*}, C_{k*} and S for a given rating Y_{ijk} simultaneously. In order to compute the updates for the SGD algorithm, we need to compute the gradients of the loss function and eventually the objective function with respect to the individual components of the model:

$$\partial_{U_{i*}} l(F_{ijk}, Y_{ijk}) = \partial_{F_{ijk}} l(F_{ijk}, Y_{ijk}) S \times_M M_{j*} \times_C C_{k*}$$
$$\partial_{M_{j*}} l(F_{ijk}, Y_{ijk}) = \partial_{F_{ijk}} l(F_{ijk}, Y_{ijk}) S \times_U U_{i*} \times_C C_{k*}$$
$$\partial_{C_{k*}} l(F_{ijk}, Y_{ijk}) = \partial_{F_{ijk}} l(F_{ijk}, Y_{ijk}) S \times_U U_{i*} \times_M M_{j*}$$
$$\partial_S l(F_{ijk}, Y_{ijk}) = \partial_{F_{ijk}} l(F_{ijk}, Y_{ijk}) U_{i*} \otimes M_{j*} \otimes C_{k*}$$

Algorithm 1 Tensor Factorization

```
Input $Y$, $d$
Initialize $U \in \mathbb{R}^{n \times d_U}$, $M \in \mathbb{R}^{m \times d_M}$, $C \in \mathbb{R}^{c \times d_C}$ and
$S \in \mathbb{R}^{d_U \times d_M \times d_C}$ with small random values.
Set $t = t_0$
while $(i,j,k)$ in observations $Y$ do
    $\eta \longleftarrow \frac{1}{\sqrt{t}}$ and $t \longleftarrow t + 1$
    $F_{ijk} = S \times_U U_{i*} \times_M M_{j*} \times_C C_{k*}$
    $U_{i*} \longleftarrow U_{i*} - \eta\lambda_U U_{i*} - \eta\partial_{U_{i*}} l(F_{ijk}, Y_{ijk})$
    $M_{j*} \longleftarrow M_{j*} - \eta\lambda_M M_{j*} - \eta\partial_{M_{j*}} l(F_{ijk}, Y_{ijk})$
    $C_{k*} \longleftarrow C_{k*} - \eta\lambda_C C_{k*} - \eta\partial_{C_{k*}} l(F_{ijk}, Y_{ijk})$
    $S \longleftarrow S - \eta\lambda_S S - \eta\partial_S l(F_{ijk}, Y_{ijk})$
end while
Output $U, M, C, S$
```

The *Multiverse Recommendation* TF method is summarized in Algorithm 1, which is easy to implement since it accesses only one row of U, M, and C at a time. In addition, it is easy to parallelize by performing several updates independently, provided that the (i,j,k) sets are all non-overlapping. Note that the algorithm scales linearly to the number of ratings K and the dimensionality of the factors d_U, d_M, d_C. Therefore, the algorithm complexity is $O(K d_U d_M d_C)$. Finally, it easily generalizes to the case of N context dimensions by adding one additional update per context variable.

3.2.4 Missing Context Information

TF also allows for an intuitive way of dealing with missing context information. Assume that we are missing the context information of a rating done by user i' on item j' $Y_{i',j'}$. Intuitively, one would like to add the rating information in the profile information of the user and the item while either not updating the information of the context profile or applying the update equally on all context profiles. There are thus two options:

- Update the $U_{i'*}$ and $M_{j'*}$ factors and skip the C and S factors update
- Update the $U_{i'*}$ and $M_{j'*}$ factors while updating all the context profiles in C, but with a step size η divided by the number of context cases c. We can then update the central tensor S using all the context cases dividing the step size by c

4. EXPERIMENTS

This section provides the experimental evaluation of the proposed *Multiverse Recommendation* TF algorithm. First, we analyze the impact of using contextual information by comparing the algorithm to standard non-context-aware MF. We then compare TF to two state-of-the art context-based collaborative filtering algorithms presented in [1, 4]. We evaluate the algorithms on two real world datasets, one of them kindly provided by Adomavicius *et al.* [1]. Moreover, in order to better understand the behavior of the methods, we use a synthetic dataset where we control the influence of the context variable.

4.1 Experimental setup

Before reporting the results, we shall now detail the experimental setup including the datasets used, the experimental protocol, and the different context-aware methods we compare against.

4.1.1 The data

We test the proposed method on six semi-synthetic data sets with ratings in a $\{1,\dots,5\}$ scale. The datasets were generated using the Yahoo! Webscope movies data set [2], which contains 221K ratings, for 11,915 movies by 7,642 users. The semi-synthetic data sets are used to analyze context-aware methods when varying the influence of the context on the user ratings. The original Yahoo! data set contains user age and gender features. We defined three age groups: users below 18, between 18 and 50, and above 50. We modified the original Yahoo! data set by replacing the gender feature with a new artificial feature $c \in \{0,1\}$ that was assigned randomly to the value 1 or 0 for each rating. This feature c is representing a contextual condition that can affect the rating. We randomly choose $\alpha * 100\%$ items from the dataset, and for these items we randomly pick $\beta * 100\%$ of the ratings to modify. We increase (or decrease) the rating value by one if $c = 1$ ($c = 0$) if the rating value was not already 5 (1). For example, if $\alpha = 0.9$ and $\beta = 0.5$, the corresponding synthetic data set has 90% of its items altered with profiles that have 50% of their ratings changed. We generated 6 semi-synthetic data sets varying $\alpha \in \{0.1, 0.5, 0.9\}$ and $\beta \in \{0.1, 0.5, 0.9\}$. Because of the way we are generating these datasets, the contextual condition is "influencing" the rating value more as α and β increase.

Data set	Users	Movies	Context Dim.	Ratings	Scale
Yahoo!	7642	11915	2	221K	1-5
Adom.	84	192	5	1464	1-13
Food	212	20	2	6360	1-5

Table 1: Data set statistics

The second dataset is derived from the one used by Adomavicius *et al.* [1]. It contains 1464 ratings by 84 users for 192 movies. The ratings were collected in a survey of college students. The students where asked to fill out a questionnaire on movies using a rating scale ranging from 1 (*hate*) to 13 (*absolutely love*). They were also queried on additional information about the context of the movie-watching experience. In our experiments we used 5 contextual features: companion, day of the week, if it was on the opening weekend, season, and year seen. Note that in our experiments we used a slightly different dataset compared to the original experiments in [1]: we used more ratings and took into account features that were not considered before, *i.e.*, the year and season when the movie was seen.

The third dataset was used by Hideki *et al.* [13] and provided by the authors. It contains food rating data from 212 users on 20 food menus. To acquire the data, the authors designed a two stage Internet questionnaire survey. The users were asked to rate the food menu while being in different levels of hunger. Moreover, some ratings were done while really experiencing the situation (*i.e.*, participants were hungry and ordered the menu) and some while imagining the situation. For example, in the virtual situation participants could be full, but should have provided a rating for a food menu imagining that they are hungry. For this data set we use two context features: the three degrees of hunger and binary feature specifying if the situation was virtual or real.

4.1.2 Evaluation Protocol

We assess the performance of the models by conducting a 5-fold cross-validation and computing the *Mean Absolute Error (MAE)*, defined as follows:

$$MAE = \frac{1}{K}\sum_{ijk}^{n,m,c} D_{ijk}|Y_{ijk} - F_{ijk}| \qquad (7)$$

where K is the total number of ratings in the test set. This measure is one of the most widely used performance measures in the recommender systems literature.

We use the *Absolute loss* function as defined in 3.2.1. It is important to note that the TF approach allows for direct optimization of the evaluation measure, *i.e.*, when using the *MAE* the ideal loss function for minimizing this measure is the Absolute loss function. However, if the error measure was set to *RMSE*, we could optimize the *least-square loss* function. For the training of the TF algorithm we use 10 epochs.

We regularize using the l_2 norm both on the matrices and the central tensor as explained in Sec. 3.2.2. Due to computational and time constrains we only conduct limited parameter search on all methods tested. For the TF algorithm, we set the regularization parameters $\lambda = \lambda_U = \lambda_M = \lambda_C$

[2] Webscope v1.0, http://research.yahoo.com/

and we thus end up with two regularization parameters λ and λ_S, the initial learning rate, and the dimensionalities of the individual components of the models d_U, d_M etc. as parameters.

All reported results are the average of a 5-fold cross-validation. We do not report on the variance of the results since it was insignificant in our experiments, and did not qualitatively influence our findings and conclusions. Moreover, all differences between TF and the other methods are statistically significant.

4.1.3 Context-based Methods in Comparison

We choose two state-of-the-art context aware collaborative filtering methods that are based on pre-filtering and compare them to N-dimensional Multiverse Recommendation based on TF.

The first one is a reduction based approach, which is based on OLAP [1], and extends classical Collaborative Filtering methods by adding contextual information to the representation of users and items. This reduction based method computes recommendations using *only* the ratings made in the same context as the target one. The exact contextual segments that optimize the prediction are searched (optimized) among those that improve the accuracy of the prediction. This is a rather computationally expensive operation as for each combination of contextual conditions, a Collaborative Filtering model needs to be trained and tested.

The second method that we use in our experiments is item splitting [4], which overcomes the computational issues of the reduction based approaches and provides a more dynamic solution. Item splitting identifies items that have significant differences in the ratings. For each of these items, it splits the ratings into two subsets, creating two new artificial items with ratings assigned to these two subsets. The split is determined by the value of one contextual variable c_j – *i.e.*, all the ratings that have been acquired in a context where the contextual feature c_j took a certain value. The method then determines if the two subsets of ratings have some (statistical significant) difference, *e.g.*, in the mean. If this is the case, the split is done and the original item in the ratings matrix is replaced by the two newly generated items.

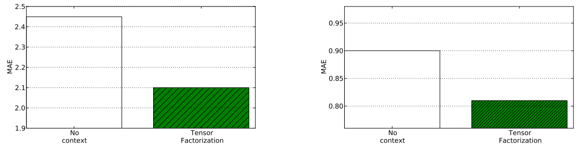
Note, that both pre-filtering methods use a standard MF approach as the main CF algorithm. Moreover, both methods exploit the tradeoff between *less* and *more relevant* data and thus increase data sparsity. This is not the case in the TF model where all the data is used to model users, item and context factors.

4.2 Results

We first conduct some experiments to assess the relevance of contextual information. In order to do so, we compare our TF method with a regular non-context-aware MF method. Then we measure the goodness of our approach by comparing it to the other context-aware methods explained in Sec. 4.1.3 on different datasets and contextual conditions.

4.2.1 Tensor vs. Matrix Factorization

We first compare the N-dimensional TF approach to standard MF. That is, we use all the available context information in the TF model while we limit the MF approach to the standard user-item-rating setting, not adding any contextual variables.



(a) Adom data

(b) Food data

Figure 2: Comparison of matrix (no context) and tensor (context) factorization on the Adom and Food data.

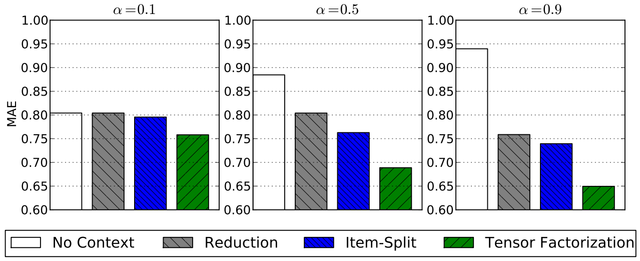


Figure 3: Comparison of context-aware methods on the Yahoo! artificial data ($\beta = 0.5$)

Figure 2 shows the results of Tensor and Matrix Factorization on the real world Adom and Food data – See Sec.4.1.1 for details on any of the used datasets. We observe that adding information in the form of context variables does make a significant (14% for the Adom and 10% for the Food data) difference in the performance. Note that when we use the Adom and Food data we end up with 7-Dimensional and 4-Dimensional TF models respectively since the Adom data contains 5 contextual variables and the Food data 2.

Figure 3 depicts the results of Tensor (in green) and Matrix Factorization (in white) on the artificial Yahoo dataset. As expected, the TF model outperforms the standard non-context aware MF method by 5% in the low context data ($\alpha = 0.1$) up to 30% in the high context case ($\alpha = 0.9$). When the context variable is not used in the Collaborative Filtering model, the contextual information acts as noise added to the data. A non-contextual model such as MF cannot distinguish between noise and a functional dependency from a hidden variable. Since we are collapsing all the information into a standard two-dimensional MF model, the method fails to model this influence. In fact, MF alone cannot exploit the additional information contained in this feature and cannot effectively deal with the influence of this variable. We observe that the stronger the influence of the context variable the higher the MAE for MF.

Note here that the training time of a *Python* implementation of the *Multiverse Recommendation* TF method on 80% of the Yahoo! data takes approximately 10 minutes on a Core 2 duo computer.

4.2.2 Comparison to Context-Aware Methods

We now compare the pre-filtering based context-aware methods to TF. Figure 3 shows a comparison of the TF method with the various context-aware CF methods. The

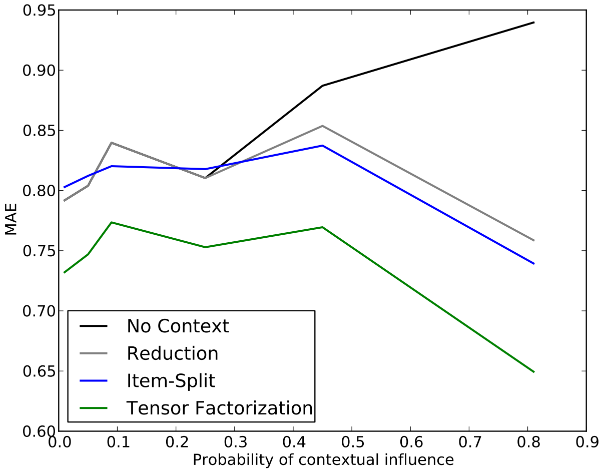
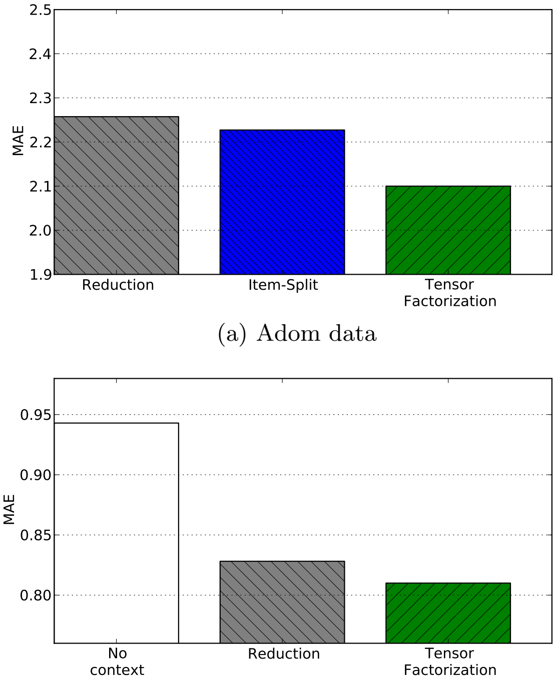


Figure 4: Evolution of MAE values for different methods with increasing influence of the context variable on the Yahoo! data.

higher the influence of the context variable, the better the TF methods performs compared to the other context-aware methods. Also note that when "context" has strong influence ($\alpha \geq 0.5, \beta \geq 0.5$), all three context-aware methods outperform the MF method from 4% in the low context case up to 12% in the high context data ($\alpha = 0.9$). The performance advantage of the context-based method increases substantially when more information is covered by the contextual feature c (see figure 4). The biggest improvement is observed when $\alpha = 0.9, \beta = 0.9$. In fact, for this data set the reduction-based approach improved MAE by 23.9%, item splitting improved by 24.2%, TF by 30.8% compared to the non-Context model.

More notably, for all 3 synthetic datasets TF uniformly outperforms all other methods. We also observe that as the influence of context increases, so does the gain when using TF when compared to both the OLAP-based approach and Item splitting. The proposed *Multiverse Recommendation* method also has the overall best performance when $\alpha = 0.9, \beta = 0.9$. It seems to efficiently exploit the linear dependency between the ratings and the contextual feature. We also observe that the performance of the non-context-aware MF model deteriorates with the growing influence of the context variable. As mentioned above, this is due to the fact that without the context variable included in the model, the influence of the context in the data acts as a source of noise that cannot be modeled by simple MF.

We carry out the second set of experiments using the described real world data – Adom and Food (see Sec.4.1.1 for more details). Figure 5 compares the same methods as in the previous experiment. The best performing method for these datasets is again TF, that outperforms both contextual pre-filtering methods and the context free method (MF) by at least 4.5%. For the Adom data we observe that TF outperforms both context-aware methods while for the food data we compare against the reduction based method which again is outperformed by the TF method by 2%.



(a) Adom data

(b) Food data

Figure 5: Comparison of context-aware methods on the Adom and Food data.

5. CONCLUSIONS

Matrix Factorization is one of the most favored approaches to Collaborative Filtering but the model is not flexible enough to add contextual dimensions in a straightforward manner. We have presented an extension of the model to N-dimensions through the use of tensors. We have adapted the generic Tensor Factorization approach to the Collaborative Filtering case by adding regularization and we have shown how this can be used to embed multiple contextual dimensions into a coherent Multiverse Recommendation model.

In the experimental results with three datasets, we obtain higher accuracy in the recommendations by taking into account contextual variables. When comparing TF to standard non-contextual MF, we measure gains that range from 5% to almost 30% both in semi-synthetic and real-world data. We have also shown that TF consistently outperforms current state-of the art context-aware recommendation approaches, with performance gains ranging from 2.5% to more than 12%. Furthermore, we have seen that the relative gain when comparing to other methods is proportional to the amount of contextual information available.

We believe that Multiverse Recommendations open up a new avenue for recommender systems and we plan to further investigate extensions of TF models for recommender systems. In particular, we are interested on investigating the use of the model to further explore temporal dependencies in standard CF settings while also dealing with implicit feedback. We also plan on exploring how multidimensional TF can be used to model non-contextual variables such as those related to content and user.

6. REFERENCES

[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactins on Information Systems*, 23(1):103–145, 2005.

[2] G. Adomavicius and A. Tuzhilin. *Recommender Systems Handbook*, chapter Context-aware Recommender Systems. Springer, 2010.

[3] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *Workshop on Context-Aware Recommender Systems (CARS 2009) in ACM Recsys 2009*, 2009.

[4] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In L. D. Bergman, A. Tuzhilin, R. Burke, A. Felfernig, and L. Schmidt-Thieme, editors, *RecSys*, pages 245–248. ACM, 2009.

[5] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proc. Intl. Conf. Machine Learning*, pages 65–72, New York, NY, 2004. ACM Press.

[6] G. Fung, O. L. Mangasarian, and A. J. Smola. Minimal kernel classifiers. *Journal of Machine Learning Research*, 3:303–321, 2002.

[7] P. J. Huber. *Robust Statistics.* John Wiley and Sons, New York, 1981.

[8] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proc. of KDD '08*, pages 426–434, 2008.

[9] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456, New York, NY, USA, 2009. ACM.

[10] L. D. Lathauwer, B. D. Moor, J. Vandewalle, and J. V. A multilinear singular value decomposition. *SIAM. J. Matrix Anal. & Appl*, 21:1253–1278, 2000.

[11] O. L. Mangasarian. Linear and nonlinear separation of patterns by linear programming. *Oper. Res.*, 13:444–452, 1965.

[12] K. Oku, S. Nakajima, J. Miyazaki, and S. Uemura. Context-aware svm for context-dependent information recommendation. In *Proceedings of the 7th international Conference on Mobile Data Management*, 2006.

[13] C. Ono, Y. Takishima, Y. Motomura, and H. Asoh. Context-aware preference model based on a study of difference between real and supposed situation data. In *UMAP '09: Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization*, pages 102–113, Berlin, Heidelberg, 2009. Springer-Verlag.

[14] C. Palmisano, A. Tuzhilin, and M. Gorgoglione. Using context to improve predictive models of customers in personalization applications. *IEEE Transactions on Knowledge and Data Engineering*, 20(11):1535 âĂŞ 1549, 2008.

[15] U. Panniello, A. Tuzhilin, M. Gorgoglione, C. Palmisano, and A. Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the ACM Recommender Systems Conference (RecSys 2009)*, pages 265–268, 2009.

[16] I. Pilaszy and D. Tikk. Recommending new movies: Even a few ratings are more valuable than metadata. In *Proc. of the 3rd ACM Conference on Recommender Systems (RecSys 2009).* ACM, 2009.

[17] S. Rendle, L. Balby Marinho, A. Nanopoulos, and L. Schmidt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 727–736, New York, NY, USA, 2009. ACM.

[18] N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, Cambridge, MA, 2005. MIT Press.

[19] N. Srebro and A. Shraibman. Rank, trace-norm and max-norm. In P. Auer and R. Meir, editors, *Proc. Annual Conf. Computational Learning Theory*, number 3559 in Lecture Notes in Artificial Intelligence, pages 545–560. Springer-Verlag, June 2005.

[20] M. Weimer, A. Karatzoglou, and M. Bruch. Maximum margin code recommendation. In *Proc. of the 3rd ACM Conference on Recommender Systems (RecSys 2009).* ACM, 2009.

[21] M. Weimer, A. Karatzoglou, and A. Smola. Improving maximum margin matrix factorization. *Machine Learning*, 72(3), September 2008.

[22] L. Xiong, X. Chen, T. Huang, and J. G. C. J. Schneider. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of SIAM Data Mining*, 2010.