Fast Context-aware Recommendations with Factorization Machines

Steffen Rendle
Social Network Analysis
University of Konstanz
78457 Konstanz, Germany
steffen.rendle@uni-konstanz.de

Zeno Gantner, Christoph Freudenthaler,
Lars Schmidt-Thieme
Information Systems and Machine Learning Lab
University of Hildesheim
31141 Hildesheim, Germany
{gantner,freudenthaler,schmidt-thieme}@ismll.de

ABSTRACT

The situation in which a choice is made is an important information for recommender systems. Context-aware recommenders take this information into account to make predictions. So far, the best performing method for context-aware rating prediction in terms of predictive accuracy is *Multiverse Recommendation* based on the Tucker tensor factorization model. However this method has two drawbacks: (1) its model complexity is exponential in the number of context variables and polynomial in the size of the factorization and (2) it only works for categorical context variables. On the other hand there is a large variety of fast but specialized recommender methods which lack the generality of context-aware methods.

We propose to apply *Factorization Machines* (FMs) to model contextual information and to provide context-aware rating predictions. This approach results in fast context-aware recommendations because the model equation of FMs can be computed in linear time both in the number of context variables and the factorization size. For learning FMs, we develop an iterative optimization method that analytically finds the least-square solution for one parameter given the other ones. Finally, we show empirically that our approach outperforms *Multiverse Recommendation* in prediction quality and runtime.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval—*Information filtering*; I.2.6 [**Artificial Intelligence**]: Learning—*Parameter Learning*

General Terms

Algorithms, Experimentation, Measurement, Performance

SIGIR'11, July 24–28, 2011, Beijing, China.
Copyright 2011 ACM 978-1-4503-0757-4/11/07 ...$10.00.

Keywords

Context-Aware Recommender System, Factorization Machine, Rating Prediction, Tensor Factorization

1. INTRODUCTION

Rating prediction in recommender systems relies primarily on the information of *how* (which rating, e.g. on a scale from 1 to 5 starts) *who* (which user) rated *what* (which item, e.g. movie, news article, or product). There are many methods that take additional data about the *who* (demographic information about the user, age, profession, gender) or the *what* (item attributes like movie genres or keywords from the product description) into account.

Besides such data about the entities involved in the rating events, there is possibly also information about the *situation* in which the rating event happens, e.g. the current location, the time, who is nearby, or the current mood of the user. Such situational information is usually called *context*. Because it is known from decision psychology that the setting and the mood of a person do influence their behavior, it is desirable to exploit context information in recommender systems. *Context-aware rating prediction* relies on the information of *how who* rated *what* in *which context* (see also figure 1).

Classical recommender system methods do not take context information into account. Some approaches perform pre- or post-processing of the data to make standard methods context-aware. While such ad-hoc solutions may work in practice, they have the shortcoming that all steps in the process need supervision and fine-tuning. Methods that integrate all kinds of input data into one model are more practical in this respect, as well as theoretically more elegant. Currently, the most flexible and strongest approach in terms of prediction accuracy is *Multiverse Recommendation* [5] that relies on Tucker decomposition and allows to work with any categorical context. However, for real-world scenarios its computational complexity is too high as it is in $O(k^m)$ where k is the dimensionality of the factorization and m the number of modes/variables involved.

In this paper, we propose a context-aware rating predictor that is based on *Factorization Machines* (FM) [14]. FMs include and can mimic the most successful approaches in recommender systems including matrix factorizion [18], SVD++ [6] or PITF [15]. We show how FMs can be applied to a wide variety of context domains including categorical, set categorical or real-valued domains. Besides this flexibility in modeling, the complexity of FMs is linear both in k and m which allows fast prediction and learning with context-aware data. For learning the model parameters of FMs, we propose a new algorithm that is based on alternating least squares (ALS). Our algorithm directly finds the optimal solution for one model parameter given all the other ones and a joint optimum is found within a few iterations. Like for stochastic gradient descent (SGD), the complexity for one iteration of our ALS algorithm is in $O(|S|\, m\, k)$ where $|S|$ is the number of training examples. The main advantage of our new ALS algorithm over SGD is that no learning rate has to be determined. This is very important in practice, because the quality of SGD learning relies largely on a good learning rate and thus an expensive search has to be done. This is not necessary for ALS.

In our experiments, we show empirically that context-aware FMs can capture context information and improve predictive accuracy. Furthermore, FMs outperform the state-of-the-art method *Multiverse Recommendation* both in prediction quality and largely in runtime.

1. In contrast to other context-aware rating predictors, FMs are easily applicable to a wide variety of contexts. We show how to generate feature vectors for context of categorical, set categorical and real-valued domains.

2. We develop a new learning algorithm for FMs that directly computes the least-square solution of a model parameter given the remaining parameters. The advantage of our ALS algorithm over SGD is that it works without a learning rate.

3. Compared to the state-of-the-art context-aware rating prediction method *Multiverse Recommendation*, FMs have a linear complexity and empirically provide a better prediction quality in much less time.

2. CONTEXT-AWARE RATING PREDICTION

We first describe the standard rating prediction task and then extend it to context-aware rating prediction. Next we show how this task can be expressed as a regression task from real-valued feature vectors under extreme sparsity. We also discuss shortly why standard regression models are not effective in this setting.

2.1 Rating prediction

Standard rating prediction can be defined as a regression task over users $U = \{u_1, u_2, \ldots\}$ and items $I = \{i_1, i_2, \ldots\}$, where a target function $y : U \times I \to \mathbb{R}$ has to be estimated. The target function represents the rating, e.g. $y(u, i)$ is the rating of user u for item i. We denote the observed part of y by $S \subset U \times I$; i.e. for all $(u, i) \in S$, we know the rating $y(u, i)$ in advance. The task of rating prediction is to estimate a function $\hat{y}$ that can predict $\hat{y}(u, i)$ for any user-item combination.

2.2 Context

In context-aware recommender systems, it is assumed that some additional information is available that influences the rating behavior. We define a context as a variable $c \in \mathcal{C}$. Examples are the mood the user was in when (s)he rated an item (e.g. $\mathcal{C} = \{happy, sad, \ldots\}$), the time at which a rating was given (e.g. $\mathcal{C} = \mathbb{R}^+$), the last items seen (e.g. $\mathcal{C} = \mathcal{P}(I)$) or the location (e.g. $\mathcal{C} = \mathbb{R}^2$).

2.3 Context-aware rating prediction

If multiple contexts $\mathcal{C}_3, \ldots, \mathcal{C}_m$ are allowed the task is to estimate the following rating function:

$$y : U \times I \times \mathcal{C}_3 \ldots \times \mathcal{C}_m \to \mathbb{R} \qquad (1)$$

Note that we start the index of the context variables with 3 because from a technical point of view, users and items can be seen as the first and second 'context'.

Figure 2 shows an example for context-aware data on the left side. There are users U in mood $\mathcal{C}_3$ watching movies I together with other users $\mathcal{C}_4$:

$$\begin{aligned} U &= \{\boldsymbol{A}lice, \boldsymbol{B}ob, \boldsymbol{C}harlie\} \\ I &= \{\boldsymbol{TI}tanic, \boldsymbol{N}otting\ \boldsymbol{H}ill, \boldsymbol{S}tar\ \boldsymbol{W}ars, \boldsymbol{S}tar\ \boldsymbol{T}rek\} \\ \mathcal{C}_3 &= \{\boldsymbol{S}ad, \boldsymbol{N}ormal, \boldsymbol{H}appy\} \\ \mathcal{C}_4 &= \mathcal{P}(U) \end{aligned}$$

The first tuple in Fig. 2 states that ***A**lice* rated ***TI**tanic* with 5 stars and that she has watched this movie with ***C**harlie* while she was ***H**appy.*

3. RELATED WORK

Most research in recommender systems focus on context-unaware methods that analyze only the user-item interaction. Here matrix factorization approaches (e.g. [18, 6]) have become very popular as they usually outperform traditional k-nearest neighbor methods (e.g. [17]). There is also research in incooperating meta-data like user or item attributes into the prediction, like Stern et al. [19] who extend a matrix factorization model. However meta-data often yields in only little or no improvement over strong baseline methods for rating prediction if enough feedback data is present [12]. The difference between such user/ item attributes and context is that attributes are attached only to either an item or user (e.g. a genre is attached to a movie) whereas context is attached to the whole rating event (e.g. the mood of a user when rating an item).

In contrast to the huge literature on standard recommender systems, there is only little research on context-aware recommender systems. The most basic approaches are contextual pre-filtering and post-filtering where a standard context-unaware recommender system is applied and the data is either preprocessed based on the context of interest before applying the recommender or the results are postprocessed [11]. Examples for pre-processing are item-splitting [3] or the multidimensional model of Adomavicius et al. [1] which is based on OLAP cubes. More sophisticated approaches use all the context and user-item information simultaneously to make predictions. Oku et al. [9] use SVMs for context-aware predictions – which have limitations in sparse applications like recommender systems as no 2-way interaction between items and users can be estimated directly [14]. Li et al. [8] suggest to see context as a user feature which is dynamic, i.e. can change. There is also some research on context-aware recommendation systems for item prediction [16], which is a ranking task instead of a regression task like the problem we are dealing with in this paper.

3.1 Multiverse Recommendation

Recently, Karatzoglou et al. [5] have proposed to apply the Tucker decomposition [20] to factorize the tensor over user, items and all categorical context variables directly. They called their approach *Multiverse Recommendation* and have shown empirically that their approach results in better prediction accuracy than item-splitting [3] and the OLAP approach [1]. In our evaluation, we will compare our *Factorization Machine* approach to *Multiverse Recommendation*.

As both our approach and *Multiverse Recommendation* are based on factorization models, we shortly recapitulate their approach and highlight the differences. Their model equation is the Tucker Decomposition, which decomposes an m-mode tensor into a smaller core tensor $\mathcal{B}$ and one factor matrix $\mathbf{V}^{(m)}$ per mode. For context-aware recommender systems, the first mode is the user, the second the item and the remaining $m-2$ modes are the context variables. The model equation can be written as:

$$\hat{y}(u, i, c_3, \ldots, c_m) := \sum_{f_1}^{k_1} \cdots \sum_{f_m}^{k_m} b_{f_1,\ldots,f_m}\, v_{u,f_1}^{(U)}\, v_{i,f_2}^{(I)} \prod_{l=3}^{m} v_{c_l,f_l}^{C_l},$$

with

$$\mathcal{B} \in \mathbb{R}^{k_1 \times \ldots \times k_m}, \quad \mathbb{V}^{(U)} \in \mathbf{R}^{|U| \times k_1}$$
$$\mathbf{V}^{(I)} \in \mathbb{R}^{|I| \times k_2}, \quad \mathbb{V}^{(C_l)} \in \mathbb{R}^{|C_l| \times k_l}.$$

The major drawback of this model is that its computational complexity is in $O(\prod_{l=1}^{m} k_l)$. Assuming factorization dimensions of equal size, i.e. $k := k_l$, this means a computational complexity of $O(k^m)$. Thus the complexity of computing one context-aware rating is exponential in the number of modes and polynomial in the number of factors. This leads to both poor learning and poor prediction runtime as soon as the number of factors growths. In Figure 5 we compare the learning runtime of *Multiverse Recommendation* empirically to our FM approach. A second limitation of *Multiverse Recommendation* in comparison to our approach is that only categorical context can be modeled – e.g. categorical set variables or real-valued variables are not possible. Finally, in the related task of item or tag recommendation (which is a ranking task), it has been shown that often it is better to factorize several lower variable interactions (e.g. pairwise ones [15]) instead of one m-ary relation (like the Tucker Decomposition). The reason is that under high sparsity, factorized pairwise relations can be estimated well but factorized m-ary relations are harder to estimate. Our FM approach follows this idea and models all nested interactions up to pairwise ones.

3.2 Attribute-aware Recommendation

In contrast to the little work on general context-aware methods, there is much more research on attribute-aware or specialized recommender systems. For example [19] or [2] present extensions of the matrix factorization model that can handle user and item attributes. There are also several works on taking time-effects into account, e.g. [7, 21]. However, all of these approaches are designed only for specific problems and cannot handle the general problem setting of context-aware recommendation that we examine in this work. For sure for specific and important problems (e.g. time-aware or attribute-aware recommendation) it is beneficial to investigate specialized methods that are supposed to be the best models for a certain problem, but on the other hand also research on general methods like context-aware recommenders is important as they offer the largest flexibility and may serve as (strong) baselines for specialized models.

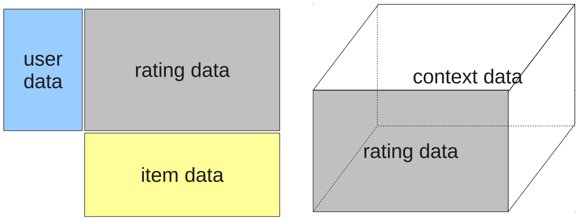


Figure 1: Attribute-aware methods can take additional information about the user or the item separately into account (left), whereas context-aware methods are more general and can analyze data that is simultaneously attached to all 'modes', i.e. the whole rating event (right).

3.3 Alternating Least Square Optimization

For the model class of matrix factorization, Bell and Koren [4] proposed an ALS method that alternates between optimizing all user factors and all item factors. As the whole factor matrix of all users (resp. items) is optimized jointly, the computation complexity is $O(k^3)$. This complexity issue of standard ALS is the reason why SGD approaches are more popular in the recommender system literature than ALS. Pilászy et al. [13] have proposed to optimize the factors within each user (resp. item) one after the other which results in an ALS algorithm constant in k, i.e. $O(k)$ because the matrix inversion is avoided. The general idea of optimizing one factor at a time is the same idea that we apply for our ALS algorithm for FMs. Both approaches [4, 13] work only for matrix factorizations and thus cannot handle any context like our proposed FMs which model all interactions. Furthermore our ALS algorithm also learns the global bias and basic 1-way effects.

4. CONTEXT-AWARE RATING PREDICTION WITH FMS

FMs are a generic model class that subsumes and can mimic several of the most successful recommender systems, among them matrix factorization [18], SVD++ [6] or PITF [15]. We shortly recapitulate the FM model and then show in detail how it can be applied to context-aware data and what happens inside an FM using such context-aware data. In the second main part, we propose a new fast alternating least square (ALS) optimization algorithm that makes FM much easier applicable compared to SGD algorithms because it works without any learning rate.

4.1 Rating Prediction with FMs

A factorization machine (FM) [14] models all interactions between pairs of variables with the target, including nested ones, by using factorized interaction parameters[1]:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i\, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \hat{w}_{i,j}\, x_i\, x_j \qquad (2)$$

where $\hat{w}_{i,j}$ are the factorized interaction parameters between pairs:

$$\hat{w}_{i,j} := \langle \mathbf{v}_i, \mathbf{v}_j \rangle = \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f} \qquad (3)$$

and the model parameters Θ that have to be estimated are:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^n, \quad \mathbf{V} \in \mathbb{R}^{n \times k} \qquad (4)$$

That means w_0 is the global bias, w_i models the interaction of the i-th variable to the target and $\hat{w}_{i,j}$ models the factorized interaction of a pair of variables with the target. Note also that unlike other factorization models like matrix factorization or PARAFAC, FMs can work with any continuous input data $\mathbf{x}$.

In [14] it has also been shown that a FM (eq. 2) can be computed very efficiently in $O(k \cdot m(\mathbf{x}))$ as it is equivalent to:

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i\, x_i + \frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f}\, x_i \right)^2 - \sum_{i=1}^{n} v_{i,f}^2\, x_i^2 \right) \qquad (5)$$

For the task of regression, the most widely used loss function is square loss. To prevent overfitting it is common to add a regularization term – usually L2. In total, we use the following regularized least square criterion for optimization:

$$\text{RLS-OPT} = \sum_{(\mathbf{x},y) \in S} (\hat{y}(\mathbf{x}) - y)^2 + \sum_{\theta \in \Theta} \lambda_{(\theta)}\, \theta^2 \qquad (6)$$

where $\lambda_{(\theta)}$ is a regularization (hyper-)parameter for the model parameter θ. In general, the regularization term can be chosen individually for each model parameter. But in practice it makes sense to use the same regularization parameters for similar model parameters – in our experiments we use: $\lambda_{(w_0)} = 0$ as there is no need to regularize the global bias; the same $\lambda_{(\mathbf{w})}$ for all parameters $w_i \in \mathbf{w}$ and the same $\lambda_{(\mathbf{V})}$ for all parameters $v_{i,f} \in \mathbf{V}$.

4.2 Context-aware FMs

A wide variety of context-aware data can be transformed into such a prediction task using real valued feature vectors $\mathbf{x} \in \mathbb{R}^n$ (see figure 2, right side for an example). We will show possible mappings $\mathbf{z} : \mathcal{C} \to \mathbb{R}^{n_z}$ for different kind of domains:

- **Categorical domain**: A categorical variable domain $\mathcal{C}$ like users U, items I, or mood can be transformed into a real-valued vector $\mathbf{z}$ using one indicator variable per categorical level. E.g. in our example the mood domain $\mathcal{C}_3$ has three states, so we can express ***A****lice*'s mood ***H****appy* with the vector $\mathbf{z}$(***H****appy*) $= (0, 0, 1)$, or if she is in a ***S****ad* context with: $\mathbf{z}$(***S****ad*) $= (1, 0, 0)$.

- **Categorical set domain**: if the realization of a variable can be sets of categorical variables (e.g. ***A****lice* has watched ***S****tar* ***W****ars* with ***B****ob* and ***C****harlie*), the transformation can be made by using one indicator variable per context level. E.g. $\mathbf{z}$({***B****ob*, ***C****harlie*}) $= (0, 0.5, 0.5)$; here we suggest to normalized $\mathbf{z}$ for non-empty context values such that all vectors sum up to 1. This makes sure that all vectors have the same weight which is often desirable.

- **Real valued domains**: If a domain is already a number (e.g. $\mathcal{C} \subseteq \mathbb{R}$), we can directly use the real number as feature, e.g. $\mathbf{z}(3.141) = 3.141$.

The final feature vector $\mathbf{x}$ can be obtained by concatenating the single mappings:

$$\mathbf{x}(u, i, c_3, \ldots, c_m) = (\mathbf{z}_1(u), \mathbf{z}_2(i), \mathbf{z}_3(c_3), \ldots, \mathbf{z}_m(c_m)) \qquad (7)$$

This data vectors $\mathbf{x}$ are then the input for the FM (eq. 2).

Example.

Figure 2 shows a complete example how to transform the context-aware data of section 2.3 into a prediction problem from real-valued features. The example contains the categorical domains user U, item I and mood $\mathcal{C}_3$ as well as a set-categorical domain, the friends the movie has been watched with, $\mathcal{C}_4$. To get an insight of how an FM will work, we investigate shortly the model equation of an FM applied to this data[2]. When we look at the feature vector $\mathbf{x}$ of our example, one can see that most of them are 0 and thus the FM model equation can be rewritten as:

$$\begin{aligned} \hat{y}(\mathbf{x}(u, i, c_3, c_4)) = {} & w_0 + w_i + w_u + w_{c_3} + \sum_{t \in c_4} x_t\, w_t \\ & + \langle \mathbf{v}_i, \mathbf{v}_u \rangle + \langle \mathbf{v}_i, \mathbf{v}_{c_3} \rangle + \langle \mathbf{v}_i, \sum_{t \in c_4} x_t\, \mathbf{v}_t \rangle \\ & + \langle \mathbf{v}_u, \mathbf{v}_{c_3} \rangle + \langle \mathbf{v}_u, \sum_{t \in c_4} x_t\, \mathbf{v}_t \rangle + \langle \mathbf{v}_{c_3}, \sum_{t \in c_4} x_t\, \mathbf{v}_t \rangle \end{aligned}$$

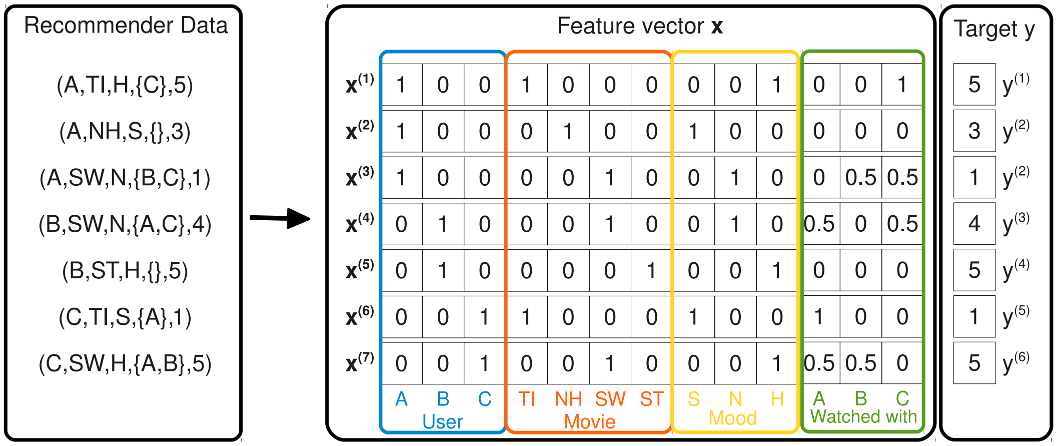
That means the FM model contains the bias term for the individual item i, user u, mood c_3 and the average bias of the co-watchers c_4. Furthermore it factorizes all pairwise interactions between these four variables. Comparing the FM model with the standard matrix factorization model (e.g. [18]) for the user u and the item i, one can see that the FM contains also exactly this factorization: $\langle \mathbf{v}_i, \mathbf{v}_u \rangle$. Additionally, it factorizes all pairwise interactions with all context variables. This shows how FMs automatically include one of the best performing recommender system models, the matrix factorization model.

4.3 Fast Learning

As the model equation of FMs can be calculated in linear time (see eq. 5), it is straightforward to develop a stochastic gradient descend (SGD) optimization algorithm for a variety of loss functions. However SGD requires to find a good learning rate which is big enough to have convergence after a reasonable amount of iterations and small enough that the gradient steps are made towards the minimum which is especially important in latter iteration stages. In the following,

[1] We restrict our discussion to 2-way FMs ($d = 2$).

[2] Please note that we do this analysis only to explain what happens implicitly in the FM – nothing has to be done explicitly when applying an FM. The end-user (domain expert) has only to specify the input features (i.e. the $\mathbf{x}$ vector) and run a generic FM tool with this data.



Recommender Data

(A,TI,H,{C},5)
(A,NH,S,{},3)
(A,SW,N,{B,C},1)
(B,SW,N,{A,C},4)
(B,ST,H,{},5)
(C,TI,S,{A},1)
(C,SW,H,{A,B},5)

Feature vector **x**

	User			Movie				Mood			Watched with			Target y	
	A	B	C	TI	NH	SW	ST	S	N	H	A	B	C		
$\mathbf{x}^{(1)}$	1	0	0	1	0	0	0	0	0	1	0	0	1	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	0	1	0	0	1	0	0	0	0	0	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	0	0	1	0	0	1	0	0	0.5	0.5	1	$y^{(2)}$
$\mathbf{x}^{(4)}$	0	1	0	0	0	1	0	0	1	0	0.5	0	0.5	4	$y^{(3)}$
$\mathbf{x}^{(5)}$	0	1	0	0	0	0	1	0	0	1	0	0	0	5	$y^{(4)}$
$\mathbf{x}^{(6)}$	0	0	1	1	0	0	0	1	0	0	1	0	0	1	$y^{(5)}$
$\mathbf{x}^{(7)}$	0	0	1	0	0	1	0	0	0	1	0.5	0.5	0	5	$y^{(6)}$

Figure 2: Context-aware recommendation data (left side) is transformed into a prediction problem from real-valued features (right side) by encoding the categorical and set categorical variables (left side) with indicator variables (right side). Here in the feature vector x, the first three values indicate the user, the next four ones the movie, the next three ones the mood and the last three ones the other users a movie has been watched with.

we suggest a new alternating least square (ALS) learning algorithm that finds the optimal value for a model parameter given the remaining ones.

4.3.1 *Alternating Least Square*

The analytical least-square solution for a model parameter can be found for FMs because they are linear functions (in each model parameter) and there is an analytic solution for every linear function. We will show this in the following two lemmas.

LEMMA 1 (LINEARITY IN θ). *A FM is a linear function with respect to every single model parameter $\theta \in \Theta$ and thus can be reexpressed as:*

$$\hat{y}(\mathbf{x}|\theta) = g_{(\theta)}(\mathbf{x}) + \theta\, h_{(\theta)}(\mathbf{x}) \tag{8}$$

where $g_{(\theta)}$ and $h_{(\theta)}$ are independent of the value[3] *of the parameter θ.*

PROOF. We proof this by stating g and h explicitly for the model parameters. For the global bias w_0, the FM can be rewritten as:

$$\hat{y}(\mathbf{x}|w_0) = w_0 \underbrace{1}_{h_{(w_0)}(\mathbf{x})} \underbrace{+ \sum_{i=1}^{n} w_i\, x_i + \sum_{i=1}^{n}\sum_{j=i+1}^{n} \hat{w}_{i,j}\, x_i\, x_j}_{g_{(w_0)}(\mathbf{x})}$$

for w_l:

$$\hat{y}(\mathbf{x}|w_l) = w_l \underbrace{x_l}_{h_{(w_l)}(\mathbf{x})} \underbrace{+ w_0 + \sum_{i=1, i\neq l}^{n} w_i\, x_i + \sum_{i=1}^{n}\sum_{j=i+1}^{n} \hat{w}_{i,j}\, x_i\, x_j}_{g_{(w_l)}(\mathbf{x})}$$

And for the factorized 2-way interactions $v_{l,f}$:

$$\hat{y}(\mathbf{x}|v_{l,f}) := v_{l,f}\, \overbrace{x_l \sum_{i=1, i\neq l} v_{i,f}\, x_i}^{h_{(v_{l,f})}(\mathbf{x})}$$
$$\underbrace{+ w_0 + \sum_{i=1}^{n} w_i\, x_i + \sum_{i=1}^{n}\sum_{j=i+1}^{n} \sum_{\substack{f'=1\\ (f'\neq f)\vee(l\notin\{i,j\})}}^{k} v_{i,f'}\, v_{j,f'}\, x_i\, x_j}_{g_{(v_{l,f})}(\mathbf{x})}$$

□

REMARK 1. *For linear decomposable functions like $\hat{y}$ of a FM, the function $h_{(\theta)}(x)$ is the gradient of $\hat{y}$ with respect to θ:*

$$h_{(\theta)}(\mathbf{x}) = \frac{\partial}{\partial\theta}\hat{y}(\mathbf{x}|\theta) \tag{9}$$

Differentiating $\hat{y}$ leads directly to this result.

LEMMA 2 (OPTIMAL VALUE FOR θ). *The regularized least-square solution of a single parameter θ for a linear model $\hat{y}(\mathbf{x}|\theta)$ is:*

$$\theta = -\frac{\sum_{(\mathbf{x},y)\in S} \left(g_{(\theta)}(\mathbf{x}) - y\right) h_{(\theta)}(\mathbf{x})}{\sum_{(\mathbf{x},y)\in S} h^2_{(\theta)}(\mathbf{x}) + \lambda_{(\theta)}} \tag{10}$$

PROOF. To gain the regularized least-square solution analytically, the first derivative of the optimization criterion (eq. 6) w.r.t. θ has to be found:

$$\frac{\partial}{\partial\theta}\text{RLS-OPT} = \sum_{(\mathbf{x},y)\in S} 2\,\left(\hat{y}(\mathbf{x}) - y\right)\, h_{(\theta)}(\mathbf{x}) + 2\,\lambda_{(\theta)}\,\theta$$

[3] The functions g and h are indexed with the **name** of the parameter θ because their form depends on the variable θ (e.g. is different for w_0 and w_l) but it does not depend on the **value** of θ.

The minimum is where this derivative is 0:

$$\sum_{(\mathbf{x},y)\in S} 2\,\left(\hat{y}(\mathbf{x})-y\right)\,h_{(\theta)}(\mathbf{x}) + 2\,\lambda_{(\theta)}\,\theta = 0$$

$$\Leftrightarrow \sum_{(\mathbf{x},y)\in S} \left(g_{(\theta)}(\mathbf{x})-y\right)\,h_{(\theta)}(\mathbf{x}) + \sum_{(\mathbf{x},y)\in S} \theta\,h^2_{(\theta)}(\mathbf{x}) + \lambda_{(\theta)}\,\theta = 0$$

$$\Leftrightarrow \theta = -\frac{\sum_{(\mathbf{x},y)\in S}\left(g_{(\theta)}(\mathbf{x})-y\right)h_{(\theta)}(\mathbf{x})}{\sum_{(\mathbf{x},y)\in S} h^2_{(\theta)}(\mathbf{x}) + \lambda_{(\theta)}}$$

□

4.3.2 Learning Algorithm

Lemma 2 with the gradients $h_{(\theta)}$ (see remark 1 and eq. (10)) allows to analytically find the optimal value of each model parameter θ of the FM given the remaining model parameters. A joint optimum of all model parameters Θ can be found iteratively by calculating the optimum of each model parameter one after another and repeating this several times. In figure 3 such an algorithm is sketched – the meaning and purpose of e and q will be explained later. First the model parameters are initialized, where the 0- and 1-way interactions (w_0 and w_l) can be initialized with 0 and the factorization parameters with small 0-centered random values. In the main loop the parameters are optimized one after the other. The idea here is to optimize first lower interactions and then higher ones because for lower interactions more data is observed and thus their estimates are more reliable. Within the factors of the 2-way interactions, first all features of the first factor dimension are optimized, then the features of the second factor dimension, etc. This allows the f-th factor to find the residuals for the 1st to $(f-1)$th factor dimensions. This optimization main loop us repeated several times to converge to the joint optimum of all model parameters.

4.3.3 Fast Calculation

Straightforward computation of the optimum for each parameter with eq. (10) would mean to calculate g and h for each training example and in each parameter update. Now, we show how to calculate the values efficiently for FMs. This involves three improvements: (1) precomputing error terms, (2) precomputing h-terms for 2-way interactions and (3) using the sparsity in S. In total, this will lead to an update algorithm where a full iteration over all parameters is in $O(|S|\,\overline{m}_S\,k)$ – i.e. linear in the number of non-zero elements in the whole dataset and the number of factors.

Precomputing error terms.

The first bottleneck in updating the parameter θ with eq. (10) is calculating $(g_{(\theta)}(\mathbf{x})-y)$ for each training case $(\mathbf{x},y)\in S$. Obviously, a trivial computation of this is in $O(m(\mathbf{x})\,k)$.

Now we will show, how to compute this in constant time $O(1)$ if the error is known. Lets define for each training case, the error[4] $e(\mathbf{x},y|\Theta)$ of the model given the model parameter as:

$$e(\mathbf{x},y|\Theta) := \hat{y}(x|\Theta) - y \qquad (11)$$

This allows to rewrite:

$$g_{(\theta)}(\mathbf{x}) - y = e(\mathbf{x},y|\Theta) - \theta\,h_{(\theta)}(\mathbf{x}) \qquad (12)$$

```
procedure LearnALS($S$)
    $w_0 \leftarrow 0$                                  ▷ Initialize the model parameters
    $\mathbf{w} \leftarrow (0,\ldots,0)$
    $\mathbf{V} \sim \mathcal{N}(0,\sigma)$
    for $(\mathbf{x},y) \in S$ do                       ▷ Precompute $e$ and $q$
        $e(\mathbf{x},y|\Theta) \leftarrow \hat{y}(\mathbf{x},y) - y$
        for $f \in \{1,\ldots,k\}$ do
            $q(\mathbf{x},f|\Theta) \leftarrow \sum_{i=1}^{n} v_{i,f}\,x_i$
        end for
    end for
    repeat                                              ▷ Main optimization loop
        $w_0^* \leftarrow -\frac{\sum_{(\mathbf{x},y)\in S}(e(\mathbf{x},y|\Theta)-w_0)}{|S|+\lambda_{(w_0)}}$      ▷ global bias
        $e(\mathbf{x},y|\Theta^*) \leftarrow e(\mathbf{x},y|\Theta) + (w_0^* - w_0)$
        $w_0 \leftarrow w_0^*$
        for $l \in \{1,\ldots,n\}$ do                   ▷ 1-way interactions
            $w_l \leftarrow -\frac{\sum_{(\mathbf{x},y)\in S}(e(\mathbf{x},y|\Theta)-w_l\,x_l)x_l}{\sum_{(\mathbf{x},y)\in S} x_l^2+\lambda_{(w_l)}}$
            $e(\mathbf{x},y|\Theta^*) \leftarrow e(\mathbf{x},y|\Theta) + (w_l^* - w_l)\,x_l$
            $w_l \leftarrow w_l^*$
        end for
        for $f \in \{1,\ldots,k\}$ do                   ▷ 2-way interactions
            for $l \in \{1,\ldots,n\}$ do
                $v_{l,f}^* \leftarrow -\frac{\sum_{(\mathbf{x},y)\in S}\left(e(\mathbf{x},y|\Theta)-v_{l,f}\,h_{(v_{l,f})}(\mathbf{x})\right)h_{(v_{l,f})}(\mathbf{x})}{\sum_{(\mathbf{x},y)\in S} h^2_{(v_{l,f})}(\mathbf{x})+\lambda_{(v_{l,f})}}$
                $e(\mathbf{x},y|\Theta^*) \leftarrow e(\mathbf{x},y|\Theta) + (v_{l,f}^* - v_{l,f})\,x_l$
                $q(\mathbf{x},f|\Theta^*) \leftarrow q(\mathbf{x},f|\Theta) + (v_{l,f}^* - v_{l,f})\,x_l$
                $v_{l,f} \leftarrow v_{l,f}^*$
            end for
        end for
    until stopping criterion is met
    return $w_0, \mathbf{w}, \mathbf{V}$
end procedure
```

Figure 3: Alternating least algorithm that optimizes the model parameters w_0, w and V for least-square in $O(|S|\,\overline{m}_{|S|}\,k)$ time (see section 4.3.3) where $|S|$ are the number of training examples and $\overline{m}_{|S|}$ the average number of non-zero elements in an input vector x.

Which can be used in the nominator of eq. (10). We store the error in a vector $\mathbf{e}\in\mathbb{R}^{|S|}$ over all training examples and precompute it at the beginning. After changing the value of a model parameter from θ to θ^*, the error also changes. This change on the error can be computed analytically by:

$$e(x,y|\Theta^*) = e(x,y|\Theta) + (\theta^* - \theta)\,h_{(\theta)}(\mathbf{x}) \qquad (13)$$

where Θ^* is the set of all model parameters where only the value of θ has changed to θ^*.

Precomputing h-terms.

After storing error terms, the computation complexity only depends on the complexity of the $h_{(\theta)}$-functions. For the parameters w_0 and w_i, the complexity of $h_{(\theta)}$ is constant, thus also computing the terms within the sums of the nominator and denominator as well as the error update is in constant time. But for the factorized parameters, computing h contains a loop over all variables. We will show now how to compute this in constant time. First, we can

[4] Actually, e is not the error, but e^2 is. Nevertheless, we use the term *error* for convenience.

reformulate $h_{(v_{l,f})}$ as:

$$h_{(v_{l,f})}(\mathbf{x}) = x_l \sum_{i=1}^{n} v_{i,f}\, x_i - x_l^2\, v_{l,f}$$

$$= x_l\, q(\mathbf{x}, f|\Theta) - x_l^2\, v_{l,f} \qquad (14)$$

$$(15)$$

with

$$q(\mathbf{x}, f|\Theta) := \sum_{i=1}^{n} v_{i,f}\, x_i \qquad (16)$$

This term q is independent of l and can be precomputed for each training case and factor in a matrix $\mathbf{Q} \in \mathbb{R}^{|S| \times k}$. With a precomputation of the q-terms, the h-function can be computed in constant time with eq. (14). When updating a parameter $v_{l,f}$ to $v^*_{l,f}$, we have to update the corresponding q-term as well. This can be done in constant time with:

$$q(\mathbf{x}, f|\Theta^*) = q(\mathbf{x}, f|\Theta) + (v^*_{l,f} - v_{l,f})\, x_l \qquad (17)$$

Again Θ^* are the new parameters after the value of $v_{l,f}$ changed to $v^*_{l,f}$, while the rest of the parameters keep unchanged.

Sparsity in S.
With the two enhancements described so far, the computation of each term within the sums of the nominator and denominator of the update rules eq. (10) can be calculated in constant time. Now, we investigate the overall complexity. First, for the w_0 parameter, the optimization complexity is $O(|S|)$. Secondly, for each update (eq. (10)) of the 1- and 2-way interactions of the l-th parameter, one has to loop only over training examples where h is nonzero – these are the training examples where $x_l \neq 0$. Thus the complexity for updating a 1-way or 2-way parameter is in $O(\overline{m}_l)$[5]. In total there are n 1-way and $n\,k$ 2-way parameters, so the complexity for one whole iteration (i.e. updating all parameters) is $O(\overline{m}_l\, n\, k) = O(|S|\, \overline{m}_{|S|}\, k)$.

5. EVALUATION

In this section, we empirically investigate if the greater model flexibility and better runtime of FMs comes to the price of less prediction quality compared to the state-of-the-art context-aware method *Multiverse Recommendation*. Furthermore we want to examine the sensitivity of SGD to the choice of the learning rate and if our ALS optimization can successfully work without this hyperparameter.

5.1 Methodology

Datasets. Even though the problem of context-aware rating prediction is highly relevant in practice, there are only a few publicly available datasets: the *Food* dataset [10], the *Adom.* dataset [1] as well as the *Yahoo! Webscope* dataset[6] enriched with context-aware information by [5]. The *Food* dataset contains 6360 ratings (1 to 5 stars) by 212 users for 20 menu items where one context variable captures if the situation in which the user rates is virtual or real (i.e. if (s)he imagines to be hungry or (s)he really is) and the second one how hungry the user is. The *Adom.* dataset contains 1524 rating events (1 to 15 stars) for movies with five context variables about companion, the weekday and other time information. For the *Yahoo! Webscope* dataset (221367 rating events) we follow [5] and apply their method to generate datasets with an increasing number of dependency of the target on the context – we generate 9 datasets for $\alpha = \beta \in \{0.1, \ldots, 0.9\}$.

Methods. We compare context-aware FMs to *Multiverse Recommendation.* Note that *Multiverse Recommendation* has been shown to outperform other context-aware recommender systems on the *Yahoo! webscope*, *Food* and *Adom.* dataset [5]. As context-unaware baseline we use FMs (*FM (nocontext)*) where only the user and item variables are used generating the feature vectors which is equivalent to matrix factorization with bias terms (see [14] for a proof) which is one of the strongest context-**un**aware recommender algorithms. All models are optimized for regularized least-squares (eq. 6). The FMs are optimized with our proposed ALS algorithm. *Multiverse Recommendation* is learned with an SGD algorithm similar to the one proposed in [5].

Protocol. We remove a 5% sample from each dataset which is used as validation set for tuning the hyperparameters for optimal MAE. After hyperparameter search, we make a 5-fold cross validation on the remaining 95% of the dataset – i.e. the validation set for hyperparameter tuning is not used any more. We report the mean RMSE and MAE over the 5 experiments. All methods are implemented in C++ and were run on the same hardware.

Experimental Reproducibility. Our FM implementation with ALS optimization, a *Multiverse Recommendation* implementation and the dataset generation scripts can be downloaded from our website[7].

5.2 Results

5.2.1 ALS vs. SGD Learning

In figure 4, we compare the test error of an SGD implementation to our ALS-approach on the Webscope dataset[8]. The leftmost figure shows the prediction quality on the test set after each iteration. The two plots on the right side show the minimal RMS error on the test set for several learning rates after 10 and 100 iterations. One can see that the prediction quality of SGD depends largely on the learning rate and the number of iterations that is chosen. If the SGD learning rate is chosen too large (e.g., here 0.01), the quality of ALS cannot be reached. With a small learning rate one can reach the quality of ALS – provided that the number of iterations is large enough (see rightmost graphs) and that one stops learning near the minimum (see leftmost graph) on a validation set to prevent overfitting.

This experiment shows that SGD needs a careful and time-consuming search for the learning rate. In contrast to this, ALS requires no such search because it does not have this

[5] Remind that m_l is the number of training cases in S where $x_l \neq 0$ which is usually low in our sparse setting

[6] `http://research.yahoo.com/Academic_Relations` ydata-ymovies-user-movie-ratings-content-v1_0

[7] `http://www.libfm.org/`

[8] The model is a context-aware FM with $k = 64$; the regularization parameters have been tuned individually for each learning method.

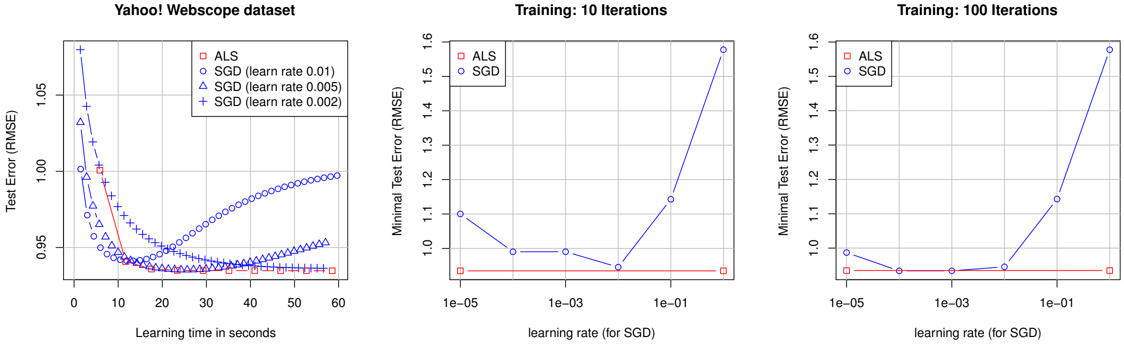


Figure 4: On the left side test error curves are plotted for three choices of learning rates. The two plots on the right show the best RMSE found after 10 and 100 iterations respectively for several choices of learning rates. This shows that for applying typical stochastic gradient descent (SGD) algorithms an appropriate learning rate has to be found. Usually an expensive search over hyperparameters is performed. Our ALS algorithm works without such a hyperparameter.

hyperparameter. With regard to prediction quality, SGD can perform as good as ALS but only with the right choice of the learning rate. This makes ALS clearly favorable in application.

5.2.2 *Runtime*

One of the main disadvantages of *Multiverse Recommendation* for practical use is that the computation complexity of the model is in $O(k^m)$ which is an issue for both learning and prediction. That means even for standard non-context-aware recommender systems, the runtime is quadratic in the dimensionality of the factorization k and for context-aware problems at least cubic. This makes it hard to apply for larger number of factors k. In contrast to this, the computational complexity of factorization machines is linear in both k and m: $O(k\,m)$.

In this experiment, we compare the runtime for one full iteration over all training examples of *Multiverse Recommendation* to context-aware FMs. As dataset, we used *Yahoo! Webscope* with $m = 4$. To show the polynomial growth of the runtime with an increasing number of the dimensionality factorization, we run both models with $k \in \{1, 2, 4, 8, 16, 32, 64, 128\}$ – for *Multiverse Recommendation* the maximal k we use was 32. In figure 5, the results are reported. These empirical results match to the theoretical complexity analysis. The runtime of the FM is linear and one full iteration over all training examples can be made for example in about 11 seconds (for $k = 128$) whereas the complexity of *Multiverse Recommendation* growths with k^4 such that the runtime for $k = 16$ is already 30 minutes and for $k = 32$ about 8 hours.

5.2.3 *Prediction Quality*

Finally, we want to investigate if the flexibility and fast runtime of context-aware FMs comes to the price of poor prediction quality compared to the *Multiverse Recommendation* method. Therefore we compare the prediction quality on the *Food*, *Adom* and *Webscope* dataset[9]. Figure 6 shows that context-aware FMs and *Multiverse Recommendation* have comparable prediction quality for the *Food* and *Adom.* datasets. Furthermore, both methods outperform the context-unaware method (here equivalent to a matrix factorization model).

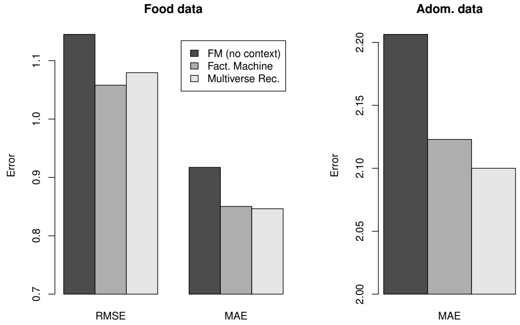


Figure 6: The context-aware methods *Multiverse Recommendation* [5] and our proposed context-aware *Factorization Machine* benefit from incoopperating the context-information into the rating prediction.

In a third experiment, we investigate the artificially enriched *Yahoo! Webscope* dataset (see [5]) with an increasing influence of context variables on the rating. Figure 7 shows that both context-aware FM and *Multiverse Recommendation* benefit from ratings that have a stronger dependence on the context. In contrast to this, the prediction quality of context-unaware FM gets worse when the rating depends stronger on the context because for context-unaware FMs the context is unobserved and thus they cannot explain this dependency.

Comparing both context-aware methods among each other, one can see that FMs throughout generate much better predictions than *Multiverse Recommendation*. E.g. for RMSE the difference is about 0.10 to 0.15 points whereas for MAE it is about 0.08 to 0.10 points. This matches to results from the related area of tag recommendation, where a pairwise interaction model (PITF) outperforms the Tucker decomposition empirically in sparse problems [15]. Note that the

[9] For the *Adom.* dataset our implementation of *Multiverse Recommendation* performed much worse than reported in [5]. Thus in favor of *Multiverse Recommendation* we report their (better) MAE value.

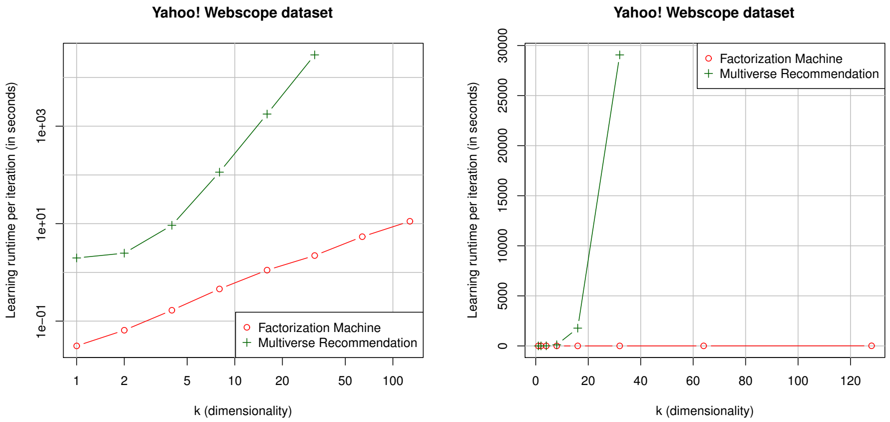


Figure 5: Learning runtime in seconds for one iteration over the whole training dataset (the left side is in log-log scale). The runtime of the *Factorization Machine* is linear in the number of latent factors k whereas for *Multiverse Recommendation* [5] it is polynomial (here $O(k^4)$). E.g. for a factorization size of $k = 16$ ($k = 32$), one iteration with the FM takes only 1.1 second (2.2 seconds), whereas for *Multiverse Recommendation* it takes about 30 minutes (8 hours).

Multiverse Recommendation model is the Tucker decomposition and the FM includes all possible pairwise interactions (like PITF).

5.2.4 Summary

Our experiments have shown that context-aware FMs are able to take contextual information into account to enhance predictions like the state-of-the-art method *Multiverse Recommendation*. In terms of runtime, FMs are linear which makes them applicable to a large dimensionality of factors, many context-variables and many observations. In contrast to this *Multiverse Recommendation* cannot handle a large number of factorization dimensions or modes. The advantage in terms of runtime is not traded in for prediction quality. Instead, on the dense datasets (*Food* and *Adom.*) the prediction quality of both methods is comparable whereas FMs outperform *Multiverse Recommendation* largely in sparse settings (*Yahoo! Webscope*). Finally, ALS optimized FMs are easily applicable as they do not require an expensive search for the learning rate like SGD optimized FMs or *Multiverse Recommendation*.

6. CONCLUSION

In this paper, we have shown how to apply factorization machines to the task of context-aware recommender systems. FMs are easily applicable to a wide variety of context by specifying only the input data. This allows them to solve scenarios where standard tensor factorization approaches cannot be applied (e.g. categorical set domains, real-valued domains). We have developed a new learning algorithm for FMs that analytically solves the least-square problem for each model parameter independently. This is especially helpful in practice as no learning rate has to be specified like in SGD approaches. Compared to *Multiverse Recommendation* which is the best-performing method for context-aware rating prediction so far, our approach achieves much faster runtime both in training and prediction ($O(k\,m)$ instead of $O(k^m)$) as well as a better prediction quality. To summarize, context-aware FMs combine the flexibility to be applied in many different scenarios and high prediction accuracy due to factorized interactions with fast and scalable computation due to the linear model complexity.

7. ACKNOWLEDGMENTS

We would like to thank Hideki Asoh, Gediminas Adomavicius and Alexander Tuzhilin for sharing their data sets.

8. REFERENCES

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

[2] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 19–28, New York, NY, USA, 2009. ACM.

[3] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 245–248, New York, NY, USA, 2009. ACM.

[4] R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA*, pages 43–52. IEEE Computer Society, 2007.

[5] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *RecSys '10: Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86, New York, NY, USA, 2010. ACM.

[6] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international*

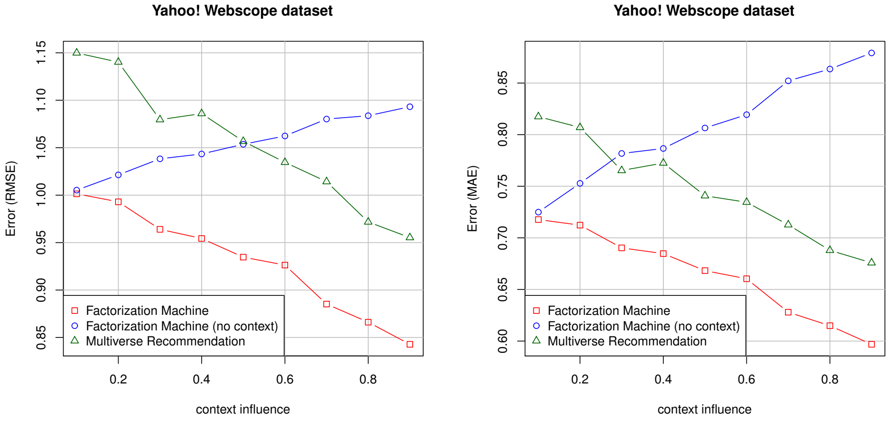


Figure 7: Prediction error with an increasing influence of the context on the rating. Context-aware FMs largely outperform *Multiverse Recommendation* [5] in quality. Secondly, context-aware FMs profit from an increasing influence of the context on the rating whereas non-contextaware FMs suffer from having no data (context) to explain the shift in the ratings.

conference on Knowledge discovery and data mining, pages 426–434, New York, NY, USA, 2008. ACM.

[7] 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.

[8] Y. Li, J. Nie, Y. Zhang, B. Wang, B. Yan, and F. Weng. Contextual recommendation based on text mining. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 692–700, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[9] K. Oku, S. Nakajima, J. Miyazaki, and S. Uemura. Context-aware svm for context-dependent information recommendation. In *MDM '06: Proceedings of the 7th International Conference on Mobile Data Management*, page 109, Washington, DC, USA, 2006. IEEE Computer Society.

[10] 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.

[11] 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 *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 265–268, New York, NY, USA, 2009. ACM.

[12] I. Pilászy and D. Tikk. Recommending new movies: even a few ratings are more valuable than metadata. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 93–100, New York, NY, USA, 2009. ACM.

[13] I. Pilászy, D. Zibriczky, and D. Tikk. Fast als-based matrix factorization for explicit and implicit feedback datasets. In *RecSys '10: Proceedings of the fourth ACM conference on Recommender systems*, pages 71–78, New York, NY, USA, 2010. ACM.

[14] S. Rendle. Factorization machines. In *Proceedings of the 10th IEEE International Conference on Data Mining*. IEEE Computer Society, 2010.

[15] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 81–90, New York, NY, USA, 2010. ACM.

[16] A. Said, S. Berkovsky, and E. W. De Luca. Putting things in context: Challenge on context-aware movie recommendation. In *CAMRa2010: Proceedings of the RecSys '10 Challenge on Context-aware Movie Recommendation*, New York, NY, USA, 2010. ACM.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM Press.

[18] N. Srebro, J. D. M. Rennie, and T. S. Jaakola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, pages 1329–1336. MIT Press, 2005.

[19] D. H. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 111–120, New York, NY, USA, 2009. ACM.

[20] L. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966.

[21] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the SIAM International Conference on Data Mining (SDM 2010)*, pages 211–222. SIAM, 2010.