Factorization Machines
Factorized Polynomial Regression Models

Christoph Freudenthaler[1], Lars Schmidt-Thieme[1] and Steffen Rendle[2]

[1] Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim, 31141 Hildesheim, Germany
[2] Social Network Analysis, University of Konstanz, 78457 Konstanz, Germany
{freudenthaler,schmidt-thieme}@ismll.uni-hildesheim.de
steffen.rendle@uni-konstanz.de

Abstract. Factorization Machines (FM) are a new model class that combines the advantages of polynomial regression models with factorization models. Like polynomial regression models, FMs are a general model class working with any real valued feature vector as input for the prediction of real-valued, ordinal or categorical dependent variables as output. However, in contrast to polynomial regression models, FMs replace two-way and all other higher order interaction effects by their factorized analogues. The factorization of higher order interactions enables efficient parameter estimation even for sparse datasets where just a few or even no observations for those higher order effects are available. Polynomial regression models without this factorization fail. This work discusses the relationship of FMs to polynomial regression models and the conceptual difference between factorizing and non-factorizing model parameters of polynomial regression models. Additionally, we show that the model equation of factorized polynomial regression models can be calculated in linear time and thus FMs can be learned efficiently. Apart from polynomial regression models, we also focus on the other origin of FMs: factorization models. We show that the standard factorization models matrix factorization and parallel factor analysis are a special case of FMs and additionally how recently proposed and successfully applied factorization models like SVD++ can be represented by FMs. The drawback of typical factorization models is that they are not applicable for general prediction tasks but work only for categorical input data. Furthermore their model equations and optimization algorithms are derived individually for each task. By knowing that FMs can mimic these models just by choosing the appropriate input data, we finally conclude that deriving a learning algorithm, e.g., Stochastic Gradient Descent (SGD) or Alternating Least Squares (ALS), for FMs once is sufficient to get the learning algorithms for all factorization models automatically, thus saving a lot of time and effort.

1 Introduction

Recommender systems are an important feature of modern websites. Especially, commercial websites benefit from a boost in customer loyalty, click-through rates

and revenue when implementing recommender systems. For example online shopping websites like Amazon give each customer personalized product recommendations that the user is probably interested in. Other examples are video portals like YouTube that recommend movies to customers.

For the majority of researchers and practitioners in the field of recommender systems, the main aim of their research has been to accurately predict ratings. This focus on rating prediction, i.e. predicting a metric, real valued variable, was particularly triggered by the Netflix prize challenge which took place from 2006 to 2009. The task of the Netflix challenge was to predict ratings of users for movies, the movie rental company Netflix offered. The challenge was structured into three different data sets: the training dataset consisting of more than 100 million ratings of almost 500 000 registered users for 17 700 different movies, a held-out validation set and a test set. The first team which was able to improve Netflix' standard recommender system by 10 % in terms of RMSE on the provided test set was awarded the grand prize of one million US-$. The Netflix prize especially popularized factorization models since the best approaches [1, 3, 2] for the Netflix challenge are based on this model class. Also on the ECML/PKDD discovery challenge[3] for tag recommendation, a factorization model based on tensor decomposition [11] has outperformed the other approaches.

Albeit their predictive power, factorization models have the disadvantage that they have to be devised individually for each problem and each set of categorical predictors. This also requires individual derivations of learning algorithms such as stochastic gradient descent [5, 6] or Bayesian inference such as Gibbs sampling [12, 16] for each factorization model. Generalizing factorization models to polynomial regression models with factorized higher-order effects, i.e. Factorization Machines (FM) [8], solves this caveat and provides in total the following advantages over hand-crafted factorization models:

- Categorical, ordinal, and metric predictors are possible
- Single derivation of a learning algorithm is valid for all emulated factorization models
- Emulation of state-of-the-art factorization models by selection of appropriate input feature vector

Compared to polynomial regression models, FMs bear other important improvements:

- Model equation can be computed in linear time
- Number of latent dimensions can be used to fine-tune the amount of expressiveness for higher-order interaction effects
- For a sufficient number of latent dimensions, FMs are as expressive as polynomial regression models
- Factorization of parameters allows for estimation of higher order interaction effects even if no observations for the interaction are available

[3] http://www.kde.cs.uni-kassel.de/ws/dc09

Thus, the general class of FMs brings together the advantages of polynomial regression models and factorization machines: general applicability and predictive accuracy. Moreover, inference in factorized polynomial regression models can be understood as an intertwined combination of a Principal Component Analysis like feature selection step and a linear regression step on the resulting latent feature space representations. Please note, that in contrast to PCA regression both steps are intertwined.

This paper is structured as follows. First, we repeat polynomial regression, go on to factorized polynomial regression, and finally introduce FMs. Next, we compare FMs to state-of-the-art factorization models and show how these models can be emulated by FMs. During this process, we discuss the conceptual properties commonalities and differences between polynomial regression, factorized polynomial regression, FMs and factorization models. Finally, in our evaluation section we will make use of previously published results on the Netflix dataset to show the equivalence of state-of-the-art factorization models to their emulating FM counterparts. This is possible since after the challenge has ended, the Netflix dataset including the ratings on test were published which made it a standard dataset for research on recommender systems. Many publications on factorization models are based on the provided train-validation-test split. Besides the equivalence study, we show results on some new factorization models which we easily developed by applying appropriate features and compare them to their non-factorized counterparts.

Please note that except for rating prediction, factorization models may also be used for item prediction, i.e. classifying items into relevant/non-relevant or sorting items according to an estimated, ordinal relevance criterion. Without loss of generality, we will deal in the evaluation section with the rating prediction case since the models for rating and item prediction are identical, just the applied loss function changes.

2 Factorization Machines - Properly Factorized Polynomial Regression

Factorization Machines can be seen as equivalent to polynomial regression models where the model parameters $\boldsymbol{\beta}$ are replaced by factorized representations thereof.

2.1 Polynomial Regression

The well-known polynomial regression model of order O for an input vector $\boldsymbol{X} \in \mathbb{R}^{p+1}$ consisting of all p predictor variables and the intercept term indicator $x_0 = 1$[4], reads for $o = 3$:

[4] Please note, that we use upper case Latin letters for denoting variables in general as well as matrices and lower case Latin letters for specific realizations of a variable. Bold printed variables are vectors or matrices.

$$y(\boldsymbol{x}|\boldsymbol{\beta}) = \beta_0 x_0 + \sum_{i=1}^{p} \beta_i x_i + \sum_{i=1}^{p} \sum_{j \geq i}^{p} \beta_{i,j} x_i x_j + \sum_{i=1}^{p} \sum_{j \geq i}^{p} \sum_{l \geq j}^{p} \beta_{i,j,l} x_i x_j x_l \quad (1)$$

Obviously, this model is an extension of the standard linear regression model. It can represent any non-linear relationship up to order O between the input parameters $\boldsymbol{X}$ and the target variable $Y \in \mathbb{R}$. A problem of polynomial regression models is the estimation of higher-order interaction effects because each interaction effect can only be learned iff these interactions are explicitly recorded in the available data set. Typically, this is not an issue for real-valued predictors since for each of these effects all training data points (except missing values) provide information about this interaction. However, for categorical predictors this is not the case because for each category linear effects are created separately, and each effect can only be estimated from observations of the related category. Training rows in which the categorical variable is not observed do not contribute to the estimation of the corresponding categorical effect. This sparsity problem for categorical variables gets even worse if higher order interaction effects of categorical variables are considered. For example, the estimation of two-way interaction effects for two categorical variables requires both categories to be observed in train. Obviously, this is a more strict requirement than for linear effects, leading to less training instances fulfilling it. This sparsity problem worsens with the order of an effect since the number of appropriate training rows further decreases.

Equation 2 gives an example of a movie rental service for a *design matrix* $\boldsymbol{X}$ containing real-valued and categorical variables. The first two columns show age of lender (in years) and age of selected movie (in months), i.e. two real-valued predictors. The last five columns show the ID of 5 different lenders, i.e. realizations of a categorical variable with five states. While higher-order interaction effects for the real-valued variables are not critical since all train rows can be used, estimation of categorical effects is problematic as the number of appropriate train rows declines with the number of categories.

$$\boldsymbol{X} = \begin{pmatrix} 25 & 5 & 1 & 0 & 0 & 0 & 0 \\ 52 & 100 & 0 & 1 & 0 & 0 & 0 \\ 34 & 7 & 0 & 0 & 1 & 0 & 0 \\ 18 & 6 & 0 & 0 & 0 & 1 & 0 \\ 55 & 60 & 0 & 0 & 0 & 0 & 1 \\ 18 & 12 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad (2)$$

Typically, the number of users is much higher. For recommender systems it is several (hundreds of) thousands. Thus, the number of predictors gets very large and the sparsity for estimating categorical linear effects gets worse. If one wants to include also the IDs for lent movies into the polynomial regression model, i.e. into the design matrix, this would finally give a matrix with at least several thousands of columns - only for the linear regression model. In the case

of the Netflix dataset the number of users is 480 000 and the number of movies is 17 700. For a polynomial regression model of order $o = 2$ estimation would become infeasible as there would be 8.5 billion predictors just for the user-item interaction. Figure 1 depicts such an extended design matrix including dummy variables for each user and for each movie, fractions summing to one for all other movies rated by the same user, a real number for the age of the movie, and an indicator for the last rated movie.

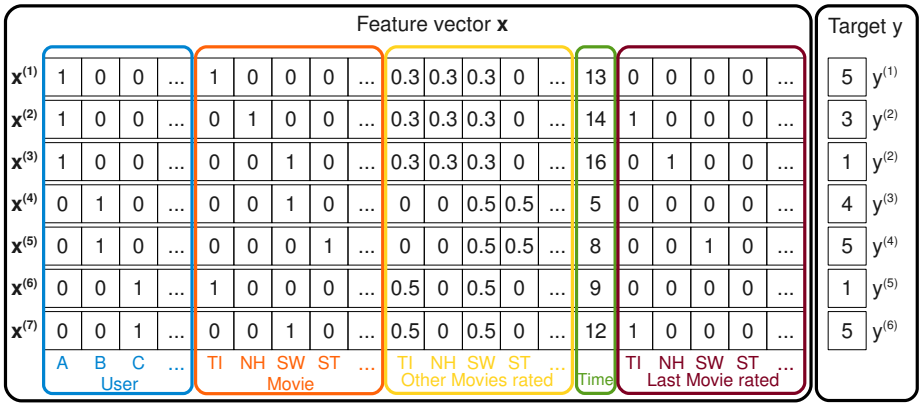

Fig. 1. Example for an extended design matrix having categorical variables user ID, item ID and last rated movie ID as well as age as real-valued predictors.

To summarize, for polynomial regression models the number of predictors is polynomial of order O in the number of predictor variables p which is particularly harmful for any categorical interaction effect because its amount of training data is *not* the training data size but the number of occurrences of that categorical interaction.

Please note, that the data sparsity problem of categorical predictors holds also true for real-valued predictors if splines and interactions of those splines are used and if the support of the spline is a proper subset of the real numbers. Besides data sparsity, another caveat of polynomial regression models is their computation time. It is polynomial in O.

These shortcomings are tackled by factorizing higher-order interaction effects.

2.2 Factorized Polynomial Regression

The rationale behind factorized polynomial regression is to remove the necessity of observing interactions for estimating interaction effects. While for polynomial regression models this necessity is active and leads to poor parameter estimates for higher-order interactions as discussed in the previous section, factorized polynomial regression models detach higher-order interaction effect estimation from

the occurrence of that interaction in train which improves parameter estimation in sparse data settings considerably. The approach of factorized polynomial regression is to replace all effects by their factorized representations. This yields for $o = 3$:

$$y(\boldsymbol{x}|\beta_0, \mathcal{V}) = \beta_0 x_0 + \sum_{i=1}^{p}\sum_{f=1}^{k_1} v_{i,f} x_i + \sum_{i=1}^{p}\sum_{j\geq i}^{p}\sum_{f=1}^{k_2} v_{i,f} v_{j,f} x_i x_j + \sum_{i=1}^{p}\sum_{j\geq i}^{p}\sum_{l\geq j}^{p}\sum_{f=1}^{k_3} v_{i,f} v_{j,f} v_{l,f} x_i x_j x_l \tag{3}$$

Since parameter factorization $\beta_i = \sum_{f=1}^{k} v_{i,f} x_i$ of one-way effects is redundant, the number of latent dimensions is fixed with $k_1 = 1$ equivalent to no factorization. In general, effects of different order have different numbers of factorization dimensions $k_1, \ldots, k_o$. This is because all effects of the same order o^* are factorized using the multi-modal parallel factor analysis (PARAFAC) factorization model of Harshman [4], which is defined as:

$$\beta_{i_1,\ldots,i_{o^*}} = \sum_{f=1}^{k_{o^*}} \prod_{j=1}^{o^*} v_{i_j,f} \tag{4}$$

Figure 2 depicts PARAFAC models for $o^* \in \{2, 3\}$.

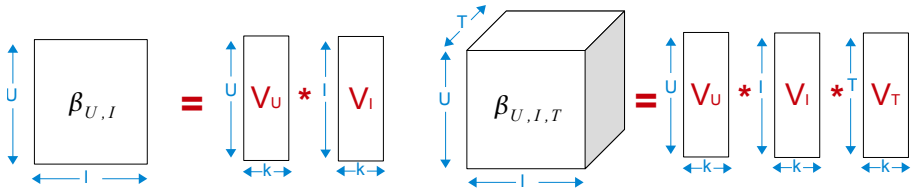

Fig. 2. PARAFAC: One parallel factor analysis model is used for each degree of interaction $o^* = 1, \ldots, o$. The left panel shows the PARAFAC model for $o^* = 2$ and the decomposition of the user-item interaction. The right panel shows PARAFAC for $o^* = 3$ and the decomposition of a user-item-point-in-time interaction.

The replacement of all model parameters $\boldsymbol{\beta}$ by factorized versions thereof replaces the original parameters $\boldsymbol{\beta}$ by a set of latent factor variables[5]

$$\mathcal{V} = \{V_{o^*} : o^* = 1, \ldots, o\} \qquad V_{o^*} \in \mathbb{R}^{p \times k_{o^*}} \ \forall o^* = 1, \ldots, o \tag{5}$$

Additionally, to emphasize that $k_1 = 1$, i.e. there is no change in the parametrization of linear effects, we change notation for the one-column matrix $V_1 \in \mathbb{R}^{p \times 1}$ to the parameter vector $\boldsymbol{w} \in \mathbb{R}^p$ and also for β_0 to w_0. This changes 3 to:

[5] Note that each matrix has p instead of $p+1$ rows as the intercept β_0 is not factorized.

$$y(\boldsymbol{x}|\mathcal{V}) = w_0 x_0 + \sum_{i=1}^{p} w_i x_i + \sum_{i=1}^{p} \sum_{j \geq i}^{p} \sum_{f=1}^{k_2} v_{i,f} v_{j,f} x_i x_j + \sum_{i=1}^{p} \sum_{j \geq i}^{p} \sum_{l \geq j}^{p} \sum_{f=1}^{k_3} v_{i,f} v_{j,f} v_{l,f} x_i x_j x_l \tag{6}$$

Thus, each predictor has for each order o^* a set of latent features. This property of factorized polynomial regression models entails several advantages:

- The number of latent dimensions can be used to fine-tune the amount of expressiveness for higher-order interaction effects.
- For a sufficient number of latent dimensions, factorized polynomial regression is as expressive as polynomial regression models.
- Factorization of parameters allows for estimation of higher order interaction effects even if no observations for the interaction are available.
- Fast computation of the model

By selecting the number of latent dimensions, the complexity of the corresponding interaction effects can be adjusted. Thus, the complexity of the model is more fine-grained. The number of latent dimensions, i.e. the optimal model complexity, is typically determined by cross-validation. Obviously, if the number of latent dimensions is large enough, the model complexity of factorized polynomial regression can be equal to polynomial regression.

Besides the adaptive model complexity, another advantage is a more efficient parameter estimation of higher-order interactions. As discussed in section 2.1 higher-order interactions suffer from data sparsity because with increasing order of an interaction effect the available data for parameter estimation decreases. However, in the case of factorized effects this correlation is eliminated, e.g., a factorized user-item interaction is learned as soon as either the corresponding item ID *or* user ID is observed whereas in the non-factorized case both item ID *and* user ID have to be observed jointly.

The last advantage of factorized polynomial regression compared to standard polynomial regression is that model predictions can be computed in linear time. Typically, the runtime complexity of calculating the prediction function 1 is polynomial in the number of parameters, i.e. $O(p^o)$. By factorization of the model parameters this complexity can be reduced to $O(p \sum_{i=1}^{o} k_o)$ by rearranging terms in equation 6 to

$$y(\boldsymbol{x}|\mathcal{V}) = w_0 x_0 + \sum_{i=1}^{p} w_i x_i + \underbrace{\sum_{f=1}^{k_2} \sum_{i=1}^{p} v_{i,f} x_i \sum_{j \geq i}^{p} v_{j,f} x_j}_{A} + \underbrace{\sum_{f=1}^{k_3} \sum_{i=1}^{p} v_{i,f} x_i \sum_{j \geq i}^{p} v_{j,f} x_j \sum_{l \geq j}^{p} v_{l,f} x_l}_{B} \tag{7}$$

and exploiting that A and B can be calculated in linear time:

$$\begin{aligned} A =& \frac{1}{2} \sum_{f=1}^{k_2} \left(\left(\sum_{i=1}^{p} v_{i,f} x_i \right)^2 + \sum_{i=1}^{p} v_{i,f}^2 x_i^2 \right) \\ B =& \sum_{f=1}^{k_3} \frac{1}{6} \left(\sum_{i=1}^{p} v_{i,f} x_i \right)^3 + \frac{5}{6} \sum_{i=1}^{p} v_{i,f}^3 x_{i,f}^3 + \frac{1}{4} \left(\left(\sum_{i=1}^{p} v_{i,f} x_{i,f} \right)^2 - \sum_{i=1}^{p} v_{i,f} x_{i,f} \right). \end{aligned} \tag{8}$$

Thus, model prediction is linear in order o and speeds up for large p. The reason for the speed up is that the nested sums can be decomposed into linear combinations of fast to compute ($O(p)$) functions of the latent factor variables. This decomposition can be derived by combinatorial analysis. For increasing order o this derivation is still possible but becomes a more and more tedious task, however, for standard applications like recommender systems it is not necessary to derive them for higher-order interactions as already two-way interactions typically suffice.

2.3 Factorization Machines - Properly Factorized Polynomial Regression

So far we have discussed factorized polynomial regression and its appealing advantages. However, up to now we have ignored an important problem of factorized polynomial regression using the PARAFAC factorization model: they are unable to model negative effects for polynomial terms like x_i^2, x_i^4,..., i.e. for terms of a polynomial with even exponents. That is why we call factorized polynomial regression *improper*. A proper solution for that issue represent Factorization Machines discussed in this section. Factorization Machines were introduced by Rendle [8]. They follow the idea of (improperly) factorized polynomial regression, thus leveraging all previously described advantages, i.e. adaptive model complexity, faster model prediction, and efficient parameter estimation in sparse data settings. However, FMs exclude those effects that take into account squares, cubes, etc. of the same predictor variable, e.g. x_i^2, x_i^3, or $x_i^2 x_j$, which leads to the following model equation for FMs:

$$y(\boldsymbol{x}|\boldsymbol{\Theta}) = w_0 x_0 + \sum_{i=1}^{p} w_i x_i + \sum_{i=1}^{p} \sum_{j>i}^{p} \sum_{f=1}^{k_2} v_{i,f} v_{j,f} x_i x_j + \sum_{i=1}^{p} \sum_{j>i}^{p} \sum_{l>j}^{p} \sum_{f=1}^{k_3} v_{i,f} v_{j,f} v_{l,f} x_i x_j x_l \tag{9}$$

This has no impact on the advantages of factorized polynomial regression models discussed in the previous section, but has two important advantages:

Properness: FMs eliminate the major drawback, the improperness of factorized polynomial regression, i.e. the inability of them to represent negative effects for polynomial functions like x_i^2, x_i^4, etc. Please note, that this does not eliminate

the ability of FMs to model this kind of higher-order interactions. If one wants to model such an effect, this can be done easily by adding the corresponding transformed predictor, e.g., x_i^2, to the set of predictors, i.e. extending the design matrix.

Multilinearity: FMs are multilinear in their parameters $\theta \in \boldsymbol{\Theta} = \{w_0, \mathcal{V}\}$, thus the model equation can be rewritten for each parameter as:

$$y(\boldsymbol{x}|\boldsymbol{\Theta}) = g\left(\boldsymbol{x}, \boldsymbol{\Theta} \setminus \theta\right) + \theta h\left(\boldsymbol{x}, \boldsymbol{\Theta} \setminus \theta\right) \tag{10}$$

where both functions $g\left(\boldsymbol{x}, \boldsymbol{\Theta} \setminus \theta\right)$ and $h\left(\boldsymbol{x}, \boldsymbol{\Theta} \setminus \theta\right)$ depend on the selected parameter θ. For a better understanding of both functions, we rewrite

$$y(\boldsymbol{x}|\Theta) - g\left(\boldsymbol{x}, \boldsymbol{\Theta} \setminus \theta\right) = \theta h\left(\boldsymbol{x}, \boldsymbol{\Theta} \setminus \theta\right) \tag{11}$$

and get the semantics of $y - g\left(\boldsymbol{x}, \boldsymbol{\Theta} \setminus \theta\right)$ as the residual error without parameter θ, i.e. $\theta = 0$ and $h\left(\boldsymbol{x}, \boldsymbol{\Theta} \setminus \theta\right)$ as the gradient of FMs w.r.t. θ. In [10] more detailed examples for both functions w.r.t. FMs are given.

This general, yet simple, dependency of all parameters $\theta \in \boldsymbol{\Theta}$ makes parameter inference much easier as exploited in [8] for stochastic gradient descent (SGD) and in [10] for alternating least squares (ALS) method.

2.4 Factorization Machines vs. Factorization Models

So far factorization models have been devised for each problem independently but no effort for generalizing all of these models has been done. This results in many different papers, e.g., [5, 6, 13, 12, 9, 11] each defining a new model and a tailored learning algorithm for each model. Having FMs this is not necessary anymore like it is not necessary to invent a polynomial regression model for each a new set of predictors. By deriving a learning algorithm for FMs all subsumed factorization models can be learned as well. In this section, we show the relation between FMs and other factorization models to demonstrate how easy state-of-the-art models can be represented as FMs. In general, the best-known factorization models are tensor factorization (like Tucker Decomposition [15], PARAFAC [4] or Matrix Factorization) that factorize an m-ary relation over m categorical variables. From a feature point of view these models require feature vectors $\mathbf{x}$ that are divided in m groups and in each group exactly one value has to be 1 and the rest 0. All existing factorization models are derived from those more general decomposition models. However, these factorization models are not as general as FMs as they are devised for categorical predictors only. Moreover, they operate with a fixed order of interactions, e.g., 2-way interaction for matrix decomposition or 3-way interactions for 3-mode tensor decomposition.

In the following, we will show according to [8] how state-of-the-art factorization models are subsumed by FMs by (i) defining a real-valued feature vector $\boldsymbol{x}$ that is created from transaction data S and (ii) analyzing what the factorization machine would exactly correspond to with such features.

Matrix Factorization Matrix factorization (MF) is one of the most studied factorization models, e.g., [14, 5, 12, 7]. It factorizes a relationship between two categorical variables, e.g., a set of users U and a set of items I:

$$y(u, i|\boldsymbol{\Theta}_{MF}) := \sum_{f=1}^{k} v_{u,f}\, v_{i,f} \tag{12}$$

with model parameters Θ_{MF}:

$$\boldsymbol{V}_U \in \mathbb{R}^{|U|\times k}, \quad \boldsymbol{V}_I \in \mathbb{R}^{|I|\times k} \tag{13}$$

An extension of this model, e.g., [5], adds bias terms on the variables:

$$y(u, i|\boldsymbol{\Theta}_{MFb}) := w_0 + w_u + w_i + \sum_{f=1}^{k} v_{u,f} v_{i,f} \tag{14}$$

with the following parameters extending $\boldsymbol{\Theta}_{MF}$ to $\boldsymbol{\Theta}_{MFb}$:

$$w_0 \in \mathbb{R}, \quad \boldsymbol{w}_U \in \mathbb{R}^{|U|}, \quad \boldsymbol{w}_I \in \mathbb{R}^{|I|}$$

This model is exactly the same as a factorization machine with $o = 2$ and two categorical predictors, i.e. users and items. Figure 3 shows the corresponding design matrix. With the corresponding feature vectors $\boldsymbol{x}$, the FM becomes:

$$\begin{aligned} y(\boldsymbol{x}|\boldsymbol{\Theta}) &= w_0 + \sum_{j=1}^{p} w_j x_j + \sum_{j=1}^{p}\sum_{l>j}^{p}\sum_{f=1}^{k} v_{j,f} v_{l,f} x_j x_l \\ &= w_0 + w_u + w_i + \sum_{f=1}^{k} v_{u,f} v_{i,f} \end{aligned} \tag{15}$$

which is the same as the bias matrix factorization model (eq. (14)). The reason is that x_j is only non-zero for user u and item i, so all other biases and interactions vanish.

Tensor Factorization Tensor factorization (TF) models generalize MF to an arbitrary number of categorical variables. Lets assume that we have a relation over m categorical variables: $X_1 \times \ldots \times X_m$, where $x_i \in \{x_{i,1}, x_{i,2}, \ldots\}$. The parallel factor analysis model (PARAFAC) [4] for a realization of an m-tuple $(x_1, \ldots, x_m)$ is defined as:

$$y(x_1, \ldots, x_m|\boldsymbol{\Theta}_{PARAFAC}) := \sum_{f=1}^{k}\prod_{i=1}^{m} v_{x_i,f} \tag{16}$$

with parameters $\boldsymbol{\Theta}_{PARAFAC}$:

$$\boldsymbol{V}_1 \in \mathbb{R}^{|X_1|\times k}, \ldots, \boldsymbol{V}_m \in \mathbb{R}^{|X_m|\times k}$$

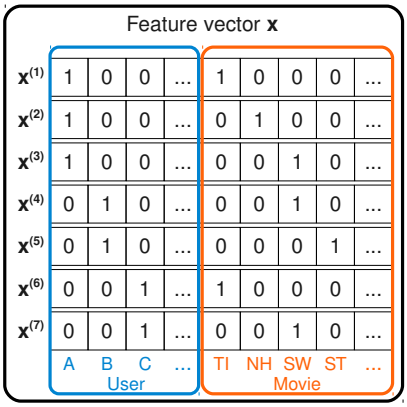

Fig. 3. Design matrix of FM emulating a biased matrix factorization.

The factorization machine ($o := m$) includes the PARAFAC model if it uses all categories as single predictors (cf. fig. 4). Then, instead of just factorizing m-way interactions like PARAFAC, FM factorizes all interaction *up to* order $o = m$:

The matrix and tensor factorization models described so far work only on 2-ary or m-ary relations over categorical variables. They cannot work, e.g., with sets of categorical variables or continuous attributes like FMs do. There are several specialized models that try to integrate other kind of information in basic tensor factorization models. Next, we describe one of them. For further mappings of succesfull factorization models into FM please refer to [8].

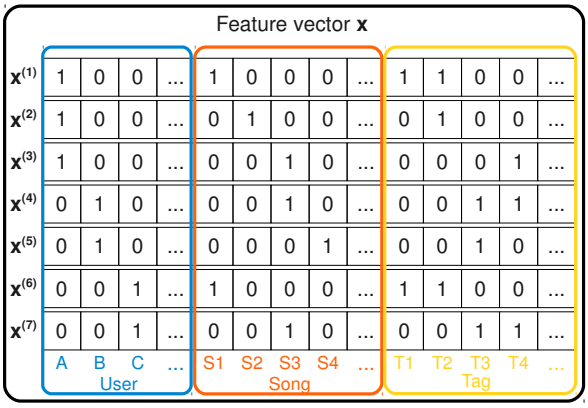

Fig. 4. Design matrix of FM emulating a mode-3 tensor. The modes are user IDs, song IDs and tag IDs. This design matrix might be used for *tag recommendation*, i.e. to predict the propensity of a user to assign a specific tag to the currently selected song.

SVD++ For the task of rating prediction, i.e. regression, Koren [5] improves the matrix factorization model to the SVD++ model. In our notation the model can be written as:

$$y(u,i|\boldsymbol{\Theta}_{SVD++}) \;:=\; w_0 \;+\; w_u \;+\; w_i \;+\; \sum_{f=1}^{k} v_{i,f} v_{u,f} \;+\; \sum_{f=1}^{k} \sum_{l \in N_u} v_{i,f} v_{l,f} \quad (17)$$

where N_u is the set of all movies the user has ever rated and the model parameters are $\boldsymbol{\Theta}_{SVD++}$:

$$w_0 \in \mathbb{R}, \quad \boldsymbol{w}_U \in \mathbb{R}^{|U|}, \quad \boldsymbol{w}_I \in \mathbb{R}^{|I|}, \quad \boldsymbol{V} \in \mathbb{R}^{(|U|+2|I|)\times k} \quad (18)$$

The information of N_u can also be encoded in the feature vector $\mathbf{x}$ using indicator variables for N_u. With the design matrix of fig. 5 the FM ($o = 2$) looks like:

$$\begin{aligned} y(\boldsymbol{x}|\boldsymbol{\Theta}) = w_0 + w_u + w_i + \sum_{f=1}^{k} v_{u,f} v_{i,f} + \frac{1}{\sqrt{|N_u|}} \sum_{l \in N_u} \sum_{f=1}^{k} v_{i,f} v_{l,f} \\ + \frac{1}{\sqrt{|N_u|}} \sum_{l \in N_u} \left(w_l + \sum_{f=1}^{k} v_{u,f} v_{l,f} \right) + \frac{1}{|N_u|} \sum_{l \in N_u} \sum_{l' \in N_u, l' > l} \sum_{f=1}^{k} v_{l,f} v_{l',f} \end{aligned} \quad (19)$$

Comparing this to the SVD++ (eq. (17)) one can see, that the factorization machine with this feature vector models exactly the same interactions as the SVD++ but the FM contains also some additional interactions between users and movies N_u as well as basic effects for the movies N_u and interactions between pairs of movies in N_u. This is a general observation when comparing FMs to factorization models. It is possible to model all factorization models but due to the completeness of FMs in terms of all interaction effects, they typically include some additional interactions.

In total, FMs and factorization models have the following differing properties:

- Standard factorization models like PARAFAC or MF are not general prediction models like factorization machines. Instead they require that the feature vector is partitioned in m parts and that in each part exactly one element is 1 and the rest 0.
- There are many proposals for specialized factorization models designed for a single task.
- Factorization machines can emulate many of the most successful factorization models (including MF, SVD++, etc.) just by feature extraction. Thus, developing one learning algorithm for FMs suffices to get learning algorithms for all subsumed factorization models automatically.

3 Experiments

After developing FMs and discussing their relationship to (factorized) polynomial regression and some factorization models we show in this section results

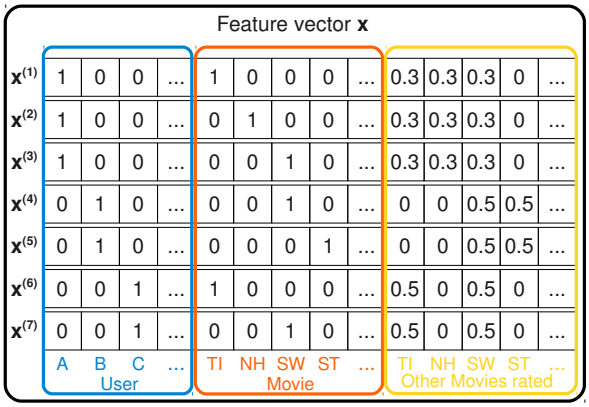

Fig. 5. Design Matrix of FM emulating SVD++.

on the Netflix dataset, thus empirically comparing FMs to some factorization models and polynomial regression models (fig. 6). For all shown FMs we set $o = 2$.

The left panel of fig. 6 shall show the ease with which different factorization models can be represented just by selecting different design matrices. By comparing them to standard factorization models (cf. 6, left panel) it becomes clear that both are equivalent. The right panel shows how FMs outperform non-factorized polynomial regression models in sparse data settings.

The statistics for the Netflix data set are as follows:

Dataset	# users	# movies	size train	size test
Netflix	480 189	17 700	100 480 507	1 408 342

3.1 FMs vs. state-of-the-art Factorization Models

The left panel of fig. 6 shows the evolution of five different models on Netflix over different sizes of the latent space k. Since all models are of order $o = 2$, the size of the latent space is determined by $k = k_2$. Results for MF and SVD++ are taken from the literature [12, 5], the other three models show FMs with different feature vectors. The names of the FMs indicate the chosen features:

- u ... user IDs
- i ... item IDs (movie IDs)
- t ... day when rating happened
- f ... natural logarithm of the number of ratings a user has made on the day the sought rating has happened

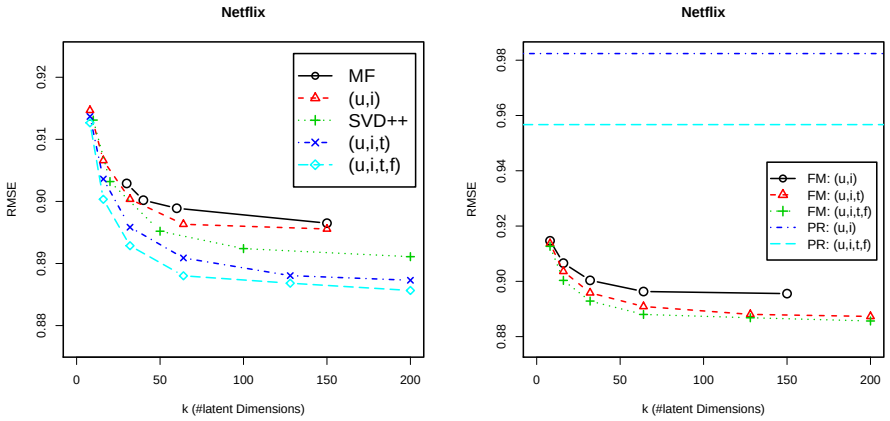

Fig. 6. Empirical comparison of several instances of FMs to selected factorization models (left panel) and to non-factorized polynomial regression models (PR, right panel).

By comparing the accuracy of MF and (u, i) the equivalence of both models can be seen. However, the FM model with features (u, i) is always slightly better than MF (eq. 13) which can be attributed to the additional bias terms for users and items (eq. 15).

The accuracies of the remaining models indicate that the additional features time t, frequency f, and co-rated items (SVD++) entail several significantly different two-way interactions of the standard user and item features with time, frequency or co-rated items, i.e. user-time, item-time, user-frequency, etc. This can be seen from the increasing gap between the (u, i)-models and the extended models as the number of latent dimensions increases. The higher the complexity, i.e. k, the better the predictive accuracy. This means that there exist quite some significant two-way interactions besides the user-item interaction which could not be captured with a low number of latent dimensions since the smaller the latent space for the modeled interaction, the lower their possible variety.

3.2 FMs vs. Polynomial Regression

Please note, the Netflix dataset is not amenable for non-factorized polynomial regression models of order $o > 1$ if the chosen predictors are categorical variables with too many categories and/or too few observations for those interactions like users and items as the corresponding estimated model parameter would suffer from too sparse data, i.e. overfitting. For instance, for the user-item interaction the Netflix dataset contains *at most* one observation of a user-item rating, but typically no observation at all. Even worse, if the rating for an item by a user is sought in test, there is no rating for that user-item pair available in train. This is a prime example for many recommender systems, e.g., online shops, where customers buy an item, e.g., movies, songs, etc., at most once. With those data

two-way or higher order interaction effects cannot be reliably estimated. Just simple linear effects are feasible.

Thus, in the case of Netflix where only user IDs, item IDs and the time of rating are available for predicting the dependent variable, polynomial regression models have to drop the user-item-interaction to avoid overfitting. Therefore, the polynomial regression model with only user ID and item ID ($PR: \ (u, i)$) as predictors in the right panel of fig. 6 is linear. The other polynomial regression model ($PR: \ (u, i, t, f)$) is able to include two-way interactions between either user ID or item ID and the time-derived covariates t and f because there are enough observations per interaction available. In the right panel of fig. 6 we compare predictive accuracies of both non-factorized polynomial regression models and their factorized siblings which are able to estimate two-way interaction effects from sparse data, i.e., the user-item interaction effect for Netflix.

The first empirical finding is, that the polynomial regression model (u, i, t, f) including time-derived predictors gets significantly better results than the simpler linear (u, i)-model. This is due to the added time but more importantly due to the added two-way effects as can be seen from the left panel of fig. 6 and from the discussion in the previous section 3.1 about the increasing gap between the user-item model and the time-enhanced models: For $k = 0$, i.e. only linear effects, the difference in accuracies is small, however, with increasing complexity of the modeled two-way interactions, i.e. k, the accuracy increases as well. Thus, in addition to a second-order user-item interaction there are other relevant second-order interaction effects in the data.

The second important finding is, that FMs extremely outperform their non-factorized siblings. The reason is that the second order interaction effect between users and items is very important for rating prediction as different users have different preferences for items which linear effects are unable to capture.

4 Conclusion

In this paper we have discussed the relationship of polynomial regression to factorized polynomial regression and Factorization Machines. There exist several advantages of factorized polynomial regression to standard polynomial regression. Especially in sparse data settings higher-order effect factorization improves the predictive accuracy.

By introducing factorized polynomial regression, particullarly FMs, the derivation of single factorization models and their learning algorithms is no longer needed. It is enough to specify the feature vectors and a preselected learning algorithm such as SGD or ALS can be used to train all factorization models. Implementations of FMs for SGD are already available[6], thus FMs are an important generalization step saving a lot of time and effort for practitioners and researchers.

[6] www.libfm.org

References

[1] R. M. Bell, Y. Koren, and C. Volinsky. The BellKor solution to the Netflix Prize, 2009.

[2] A. Töscher, M. Jahrer, and R. M. Bell. The BigChaos Solution to the Netflix Grand Prize, 2009.

[3] M. Piotte, and M. Chabbert. The Pragmatic Theory Solution to the Netflix Grand Prize, 2009.

[4] R. A. Harshman. Foundations of the PARAFAC procedure: models and conditions for an 'exploratory' multimodal factor analysis. In *UCLA Working Papers in Phonetics*, pages 1–84, volume 16, 1970.

[5] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, New York, 2008. ACM.

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

[7] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. M. Lukose, M. Scholz, and Q. Yiang. One-Class Collaborative Filtering. In *Proceedings of the IEEE International Conference on Data Mining*, pages 502–511, 2008.

[8] S. Rendle. Factorization Machines. In *Proceedings of the 10th IEEE International Conference on Data Mining*, pages 995–1000, 2010.

[9] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing Personalized Markov Chains for Next-Basket Recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 811–820, 2010, New York. ACM.

[10] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast Context-aware Recommendations with Factorization Machines. In *Proceedings of the 34th ACM SIGIR Conference on Reasearch and Development in Information Retrieval*, 2011, New York. ACM.

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

[12] R. Salakhutdinov, and A. Mnih. Bayesian Probabilistic Matrix Factorization using Markov chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[13] R. Salakhutdinov, and A. Mnih. Probabilistic Matrix Factorization. In *Proceedings of Advances in Neural Information Processing Systems*, 2008.

[14] N. Srebro, J. D. M. Rennie, and T. S. Jaakola. Maximum-Margin Matrix Factorization. In *Advances in Neural Information Processing Systems*, pages 1329–1336, 2005. MIT Press.

[15] L. R. Tucker. MSome mathematical notes on three-mode factor analysis. In *Psychometrika*, pages 279–311, volume 31, 1966.

[16] L. Xiong, X. Chen, T. Huang, J. Schneider, and J. G. Carbonell. Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization. In *Proceedings of the SIAM Conference on Data Mining*, pages 211–222, 2010.