Factorization Machines

Steffen Rendle
Department of Reasoning for Intelligence
The Institute of Scientific and Industrial Research
Osaka University, Japan
rendle@ar.sanken.osaka-u.ac.jp

***Abstract*—In this paper, we introduce Factorization Machines (FM) which are a new model class that combines the advantages of Support Vector Machines (SVM) with factorization models. Like SVMs, FMs are a general predictor working with any real valued feature vector. In contrast to SVMs, FMs model all interactions between variables using factorized parameters. Thus they are able to estimate interactions even in problems with huge sparsity (like recommender systems) where SVMs fail. We show that the model equation of FMs can be calculated in linear time and thus FMs can be optimized directly. So unlike nonlinear SVMs, a transformation in the dual form is not necessary and the model parameters can be estimated directly without the need of any support vector in the solution. We show the relationship to SVMs and the advantages of FMs for parameter estimation in sparse settings.**

On the other hand there are many different factorization models like matrix factorization, parallel factor analysis or specialized models like SVD++, PITF or FPMC. The drawback of these models is that they are not applicable for general prediction tasks but work only with special input data. Furthermore their model equations and optimization algorithms are derived individually for each task. We show that FMs can mimic these models just by specifying the input data (i.e. the feature vectors). This makes FMs easily applicable even for users without expert knowledge in factorization models.

***Index Terms*—factorization machine; sparse data; tensor factorization; support vector machine**

I. Introduction

Support Vector Machines are one of the most popular predictors in machine learning and data mining. Nevertheless in settings like collaborative filtering, SVMs play no important role and the best models are either direct applications of standard matrix/ tensor factorization models like PARAFAC [1] or specialized models using factorized parameters [2], [3], [4]. In this paper, we show that the only reason why standard SVM predictors are not successful in these tasks is that they cannot learn reliable parameters ('hyperplanes') in complex (non-linear) kernel spaces under very sparse data. On the other hand, the drawback of tensor factorization models and even more for specialized factorization models is that (1) they are not applicable to standard prediction data (e.g. a real valued feature vector in $\mathbb{R}^n$.) and (2) that specialized models are usually derived individually for a specific task requiring effort in modelling and design of a learning algorithm.

In this paper, we introduce a new predictor, the *Factorization Machine (FM)*, that is a general predictor like SVMs but is also able to estimate reliable parameters under very high sparsity. The factorization machine models all nested variable interactions (comparable to a polynomial kernel in SVM), but uses a factorized parametrization instead of a dense parametrization like in SVMs. We show that the model equation of FMs can be computed in linear time and that it depends only on a linear number of parameters. This allows direct optimization and storage of model parameters without the need of storing any training data (e.g. support vectors) for prediction. In contrast to this, non-linear SVMs are usually optimized in the dual form and computing a prediction (the model equation) depends on parts of the training data (the *support vectors*). We also show that FMs subsume many of the most successful approaches for the task of collaborative filtering including biased MF, SVD++ [2], PITF [3] and FPMC [4].

In total, the advantages of our proposed FM are:

1) FMs allow parameter estimation under very sparse data where SVMs fail.
2) FMs have linear complexity, can be optimized in the primal and do not rely on support vectors like SVMs. We show that FMs scale to large datasets like Netflix with 100 millions of training instances.
3) FMs are a general predictor that can work with any real valued feature vector. In contrast to this, other state-of-the-art factorization models work only on very restricted input data. We will show that just by defining the feature vectors of the input data, FMs can mimic state-of-the-art models like biased MF, SVD++, PITF or FPMC.

II. Prediction under Sparsity

The most common prediction task is to estimate a function $y : \mathbb{R}^n \to T$ from a real valued feature vector $\mathbf{x} \in \mathbb{R}^n$ to a target domain T (e.g. $T = \mathbb{R}$ for regression or $T = \{+, -\}$ for classification). In supervised settings, it is assumed that there is a training dataset $D = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots\}$ of examples for the target function y given. We also investigate the ranking task where the function y with target $T = \mathbb{R}$ can be used to score feature vectors $\mathbf{x}$ and sort them according to their score. Scoring functions can be learned with pairwise training data [5], where a feature tuple $(\mathbf{x}^{(A)}, \mathbf{x}^{(B)}) \in D$ means that $\mathbf{x}^{(A)}$ should be ranked higher than $\mathbf{x}^{(B)}$. As the pairwise ranking relation is antisymmetric, it is sufficient to use only positive training instances.

In this paper, we deal with problems where $\mathbf{x}$ is highly sparse, i.e. almost all of the elements x_i of a vector $\mathbf{x}$ are zero. Let $m(\mathbf{x})$ be the number of non-zero elements in the

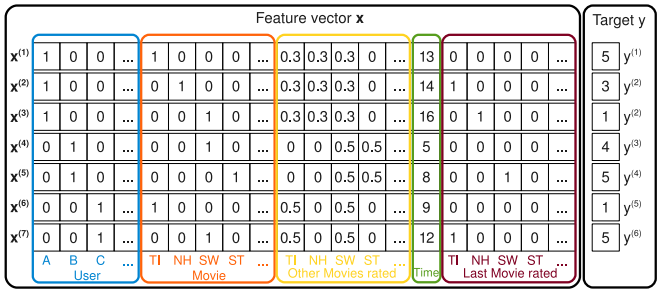

Fig. 1. Example for sparse real valued feature vectors $\mathbf{x}$ that are created from the transactions of example 1. Every row represents a feature vector $\mathbf{x}^{(i)}$ with its corresponding target $y^{(i)}$. The first 4 columns (blue) represent indicator variables for the active user; the next 5 (red) indicator variables for the active item. The next 5 columns (yellow) hold additional implicit indicators (i.e. other movies the user has rated). One feature (green) represents the time in months. The last 5 columns (brown) have indicators for the last movie the user has rated before the active one. The rightmost column is the target – here the rating.

feature vector $\mathbf{x}$ and $\overline{m}_D$ be the average number of non-zero elements $m(\mathbf{x})$ of all vectors $\mathbf{x} \in D$. Huge sparsity ($\overline{m}_D \ll n$) appears in many real-world data like feature vectors of event transactions (e.g. purchases in recommender systems) or text analysis (e.g. bag of word approach). One reason for huge sparsity is that the underlying problem deals with large categorical variable domains.

Example 1 Assume we have the transaction data of a movie review system. The system records which user $u \in U$ rates a movie (item) $i \in I$ at a certain time $t \in \mathbb{R}$ with a rating $r \in \{1, 2, 3, 4, 5\}$. Let the users U and items I be:

$$U = \{\text{Alice (A)}, \text{Bob (B)}, \text{Charlie (C)}, \ldots\}$$
$$I = \{\text{Titanic (TI)}, \text{Notting Hill (NH)}, \text{Star Wars (SW)}, \text{Star Trek (ST)}, \ldots\}$$

Let the observed data S be:

$$S = \{(\text{A}, \text{TI}, \text{2010-1}, 5), (\text{A}, \text{NH}, \text{2010-2}, 3), (\text{A}, \text{SW}, \text{2010-4}, 1), (\text{B}, \text{SW}, \text{2009-5}, 4), (\text{B}, \text{ST}, \text{2009-8}, 5), (\text{C}, \text{TI}, \text{2009-9}, 1), (\text{C}, \text{SW}, \text{2009-12}, 5)\}$$

An example for a prediction task using this data, is to estimate a function $\hat{y}$ that predicts the rating behaviour of a user for an item at a certain point in time.

Figure 1 shows one example of how feature vectors can be created from S for this task.[1] Here, first there are $|U|$ binary indicator variables (blue) that represent the active user of a transaction – there is always exactly one active user in each transaction $(u, i, t, r) \in S$, e.g. user *Alice* in the first one ($x_A^{(1)} = 1$). The next $|I|$ binary indicator variables (red) hold the active item – again there is always exactly one active item (e.g. $x_{\text{TI}}^{(1)} = 1$). The feature vectors in figure 1 also contain indicator variables (yellow) for all the other movies the user has ever rated. For each user, the variables are normalized such that they sum up to 1. E.g. *Alice* has rated *Titanic*, *Notting Hill* and *Star Wars*. Additionally the example contains a variable (green) holding the time in months starting from January, 2009. And finally the vector contains information of the last movie (brown) the user has rated before (s)he rated the active one – e.g. for $\mathbf{x}^{(2)}$, *Alice* rated *Titanic* before she rated *Notting Hill*. In section V, we show how factorization machines using such feature vectors as input data are related to specialized state-of-the-art factorization models.

We will use this example data throughout the paper for illustration. However please note that FMs are general predictors like SVMs and thus are applicable to any real valued feature vectors and are not restricted to recommender systems.

[1] To simplify readability, we will use categorical levels (e.g. *Alice (A)*) instead of numbers (e.g. 1) to identify elements in vectors wherever it makes sense (e.g. we write x_A or x_{Alice} instead of x_1).

III. Factorization Machines (FM)

In this section, we introduce factorization machines. We discuss the model equation in detail and show shortly how to apply FMs to several prediction tasks.

A. Factorization Machine Model

1) Model Equation: The model equation for a factorization machine of degree $d = 2$ is defined as:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \quad (1)$$

where the model parameters that have to be estimated are:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^n, \quad \mathbf{V} \in \mathbb{R}^{n \times k} \quad (2)$$

And $\langle \cdot, \cdot \rangle$ is the dot product of two vectors of size k:

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

A row $\mathbf{v}_i$ within $\mathbf{V}$ describes the i-th variable with k factors. $k \in \mathbb{N}_0^+$ is a hyperparameter that defines the dimensionality of the factorization.

A 2-way FM (degree $d = 2$) captures all single and pairwise interactions between variables:

- w_0 is the global bias.
- w_i models the strength of the i-th variable.
- $\hat{w}_{i,j} := \langle \mathbf{v}_i, \mathbf{v}_j \rangle$ models the interaction between the i-th and j-th variable. Instead of using an own model parameter $w_{i,j} \in \mathbb{R}$ for each interaction, the FM models the interaction by factorizing it. We will see later on, that this is the key point which allows high quality parameter estimates of higher-order interactions ($d \geq 2$) under sparsity.

2) Expressiveness: It is well known that for any positive definite matrix $\mathbf{W}$, there exists a matrix $\mathbf{V}$ such that $\mathbf{W} = \mathbf{V} \cdot \mathbf{V}^t$ provided that k is sufficiently large. This shows that a FM can express any interaction matrix $\mathbf{W}$ if k is chosen large enough. Nevertheless in sparse settings, typically a small k should be chosen because there is not enough data to estimate complex interactions $\mathbf{W}$. Restricting k – and thus the expressiveness of the FM – leads to better generalization and thus improved interaction matrices under sparsity.

3) Parameter Estimation Under Sparsity: In sparse settings, there is usually not enough data to estimate interactions between variables directly and independently. Factorization machines can estimate interactions even in these settings well because they break the independence of the interaction parameters by factorizing them. In general this means that the data for one interaction helps also to estimate the parameters for related interactions. We will make the idea more clear with an example from the data in figure 1. Assume we want to estimate the interaction between *Alice* (A) and *Star Trek* (ST) for predicting the target y (here the *rating*). Obviously, there is no case $\mathbf{x}$ in the training data where both variables x_A and x_{ST} are non-zero and thus a direct estimate would lead to no interaction ($w_{A,\text{ST}} = 0$). But with the factorized interaction parameters $\langle \mathbf{v}_A, \mathbf{v}_{\text{ST}} \rangle$ we can estimate the interaction even in this case. First of all, *Bob* and *Charlie* will have similar factor vectors $\mathbf{v}_B$ and $\mathbf{v}_C$ because both have similar interactions with *Star Wars* ($\mathbf{v}_{\text{SW}}$) for predicting ratings – i.e. $\langle \mathbf{v}_B, \mathbf{v}_{\text{SW}} \rangle$ and $\langle \mathbf{v}_C, \mathbf{v}_{\text{SW}} \rangle$ have to be similar. *Alice* ($\mathbf{v}_A$) will have a different factor vector from *Charlie* ($\mathbf{v}_C$) because she has different interactions with the factors of *Titanic* and *Star Wars* for predicting ratings. Next, the factor vectors of *Star Trek* are likely to be similar to the one of *Star Wars* because *Bob* has similar interactions for both movies for predicting y. In total, this means that the dot product (i.e. the interaction) of the factor vectors of *Alice* and *Star Trek* will be similar to the one of *Alice* and *Star Wars* – which also makes intuitively sense.

4) Computation: Next, we show how to make FMs applicable from a computational point of view. The complexity of straight forward computation of eq. (1) is in $O(k\,n^2)$ because all pairwise interactions have to be computed. But with reformulating it drops to linear runtime.

Lemma 3.1: The model equation of a factorization machine (eq. (1)) can be computed in linear time $O(k\,n)$.

Proof: Due to the factorization of the pairwise interactions, there is no model parameter that directly depends on two variables (e.g. a parameter with an index (i, j)). So the pairwise interactions can be reformulated:

$$
\begin{aligned}
&\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle\, x_i\, x_j \\
=&\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle\, x_i\, x_j - \frac{1}{2} \sum_{i=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_i \rangle\, x_i\, x_i \\
=&\frac{1}{2} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{f=1}^{k} v_{i,f}\, v_{j,f}\, x_i\, x_j - \sum_{i=1}^{n} \sum_{f=1}^{k} v_{i,f}\, v_{i,f}\, x_i\, x_i \right) \\
=&\frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f}\, x_i \right) \left(\sum_{j=1}^{n} v_{j,f}\, x_j \right) - \sum_{i=1}^{n} v_{i,f}^2\, x_i^2 \right) \\
=&\frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f}\, x_i \right)^2 - \sum_{i=1}^{n} v_{i,f}^2\, x_i^2 \right)
\end{aligned}
$$

This equation has only linear complexity in both k and n – i.e. its computation is in $O(k\,n)$. ■

Moreover, under sparsity most of the elements in $\mathbf{x}$ are 0 (i.e. $m(\mathbf{x})$ is small) and thus, the sums have only to be computed over the non-zero elements. Thus in sparse applications, the computation of the factorization machine is in $O(k\,\overline{m}_D)$ – e.g. $\overline{m}_D = 2$ for typical recommender systems like MF approaches (see section V-A).

B. Factorization Machines as Predictors

FM can be applied to a variety of prediction tasks. Among them are:

- **Regression**: $\hat{y}(\mathbf{x})$ can be used directly as the predictor and the optimization criterion is e.g. the minimal least square error on D.
- **Binary classification**: the sign of $\hat{y}(\mathbf{x})$ is used and the parameters are optimized for hinge loss or logit loss.
- **Ranking**: the vectors $\mathbf{x}$ are ordered by the score of $\hat{y}(\mathbf{x})$ and optimization is done over pairs of instance vectors $(\mathbf{x}^{(\mathbf{a})}, \mathbf{x}^{(\mathbf{b})}) \in D$ with a pairwise classification loss (e.g. like in [5]).

In all these cases, regularization terms like $L2$ are usually added to the optimization objective to prevent overfitting.

C. Learning Factorization Machines

As we have shown, FMs have a closed model equation that can be computed in linear time. Thus, the model parameters (w_0, $\mathbf{w}$ and $\mathbf{V}$) of FMs can be learned efficiently by gradient descent methods – e.g. stochastic gradient descent (SGD) – for a variety of losses, among them are square, logit or hinge loss. The gradient of the FM model is:

$$
\frac{\partial}{\partial \theta} \hat{y}(\mathbf{x}) = \begin{cases} 1, & \text{if } \theta \text{ is } w_0 \\ x_i, & \text{if } \theta \text{ is } w_i \\ x_i \sum_{j=1}^{n} v_{j,f} x_j - v_{i,f} x_i^2, & \text{if } \theta \text{ is } v_{i,f} \end{cases} \tag{4}
$$

The sum $\sum_{j=1}^{n} v_{j,f} x_j$ is independent of i and thus can be precomputed (e.g. when computing $\hat{y}(x)$). In general, each gradient can be computed in constant time $O(1)$. And all parameter updates for a case $(\mathbf{x}, y)$ can be done in $O(k\,n)$ – or $O(k\,m(\mathbf{x}))$ under sparsity.

We provide a generic implementation, LIBFM[2], that uses SGD and supports both element-wise and pairwise losses.

D. d-way Factorization Machine

The 2-way FM described so far can easily be generalized to a d-way FM:

$$
\begin{aligned}
\hat{y}(x) :=\ & w_0 + \sum_{i=1}^{n} w_i\, x_i \\
&+ \sum_{l=2}^{d} \sum_{i_1=1}^{n} \cdots \sum_{i_l=i_{l-1}+1}^{n} \left(\prod_{j=1}^{l} x_{i_j} \right) \left(\sum_{f=1}^{k_l} \prod_{j=1}^{l} v_{i_j,f}^{(l)} \right)
\end{aligned} \tag{5}
$$

[2]http://www.libfm.org

where the interaction parameters for the l-th interaction are factorized by the PARAFAC model [1] with the model parameters:

$$\mathbf{V}^{(l)} \in \mathbb{R}^{n \times k_l}, \quad k_l \in \mathbb{N}_0^+ \tag{6}$$

The straight-forward complexity for computing eq. (5) is $O(k_d\, n^d)$. But with the same arguments as in lemma 3.1, one can show that it can be computed in linear time.

E. Summary

FMs model all possible interactions between values in the feature vector $\mathbf{x}$ using factorized interactions instead of full parametrized ones. This has two main advantages:

1) The interactions between values can be estimated even under high sparsity. Especially, it is possible to generalize to unobserved interactions.
2) The number of parameters as well as the time for prediction and learning is linear. This makes direct optimization using SGD feasible and allows optimizing against a variety of loss functions.

In the remainder of this paper, we will show the relationships between factorization machines and support vector machines as well as matrix, tensor and specialized factorization models.

IV. FMs vs. SVMs

A. SVM model

The model equation of an SVM [6] can be expressed as the dot product between the transformed input $\mathbf{x}$ and model parameters $\mathbf{w}$: $\hat{y}(\mathbf{x}) = \langle \phi(\mathbf{x}), \mathbf{w} \rangle$, where ϕ is a mapping from the feature space $\mathbb{R}^n$ into a more complex space $\mathcal{F}$. The mapping ϕ is related to the kernel with:

$$K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \quad K(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle$$

In the following, we discuss the relationships of FMs and SVMs by analyzing the primal form of the SVMs[3].

1) Linear kernel: The most simple kernel is the linear kernel: $K_l(\mathbf{x}, \mathbf{z}) := 1 + \langle \mathbf{x}, \mathbf{z} \rangle$, which corresponds to the mapping $\phi(\mathbf{x}) := (1, x_1, \ldots, x_n)$. And thus the model equation of a linear SVM can be rewritten as:

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i\, x_i, \quad w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^n \tag{7}$$

It is obvious that a linear SVM (eq. (7)) is identical to a FM of degree $d = 1$ (eq. (5)).

2) Polynomial kernel: The polynomial kernel allows the SVM to model higher interactions between variables. It is defined as $K(\mathbf{x}, \mathbf{z}) := (\langle \mathbf{x}, \mathbf{z} \rangle + 1)^d$. E.g. for $d = 2$ this corresponds to the following mapping:

$$\begin{aligned}\phi(\mathbf{x}) := (&1, \sqrt{2}\, x_1, \ldots, \sqrt{2}\, x_n, x_1^2, \ldots, x_n^2, \\ &\sqrt{2}\, x_1\, x_2, \ldots, \sqrt{2}\, x_1\, x_n, \sqrt{2}\, x_2\, x_3, \ldots, \sqrt{2}\, x_{n-1}\, x_n)\end{aligned} \tag{8}$$

[3]In practice, SVMs are solved in the dual form and the mapping ϕ is not performed explicitly. Nevertheless, the primal and dual have the same solution (optimum), so all our arguments about the primal hold also for the dual form.

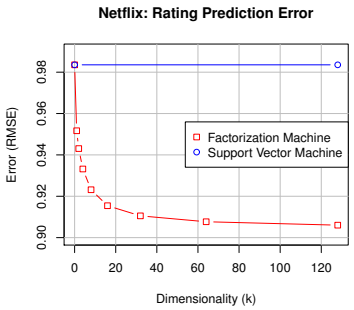

Fig. 2. FMs succeed in estimating 2-way variable interactions in very sparse problems where SVMs fail (see section III-A3 and IV-B for details.)

And so, the model equation for polynomial SVMs can be rewritten as:

$$\begin{aligned}\hat{y}(\mathbf{x}) = w_0 + \sqrt{2}\sum_{i=1}^{n} w_i\, x_i + \sum_{i=1}^{n} w_{i,i}^{(2)} x_i^2 \\ + \sqrt{2}\sum_{i=1}^{n}\sum_{j=i+1}^{n} w_{i,j}^{(2)}\, x_i\, x_j\end{aligned} \tag{9}$$

where the model parameters are:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^n, \quad \mathbf{W}^{(2)} \in \mathbb{R}^{n \times n} \text{ (symmetric matrix)}$$

Comparing a polynomial SVM (eq. (9)) to a FM (eq. (1)), one can see that both model all nested interactions up to degree $d = 2$. The main difference between SVMs and FMs is the parametrization: all interaction parameters $w_{i,j}$ of SVMs are completely independent, e.g. $w_{i,j}$ and $w_{i,l}$. In contrast to this the interaction parameters of FMs are factorized and thus $\langle \mathbf{v}_i, \mathbf{v}_j \rangle$ and $\langle \mathbf{v}_i, \mathbf{v}_l \rangle$ depend on each other as they overlap and share parameters (here $\mathbf{v}_i$).

B. Parameter Estimation Under Sparsity

In the following, we will show why linear and polynomial SVMs fail for very sparse problems. We show this for the example of collaborative filtering with user and item indicator variables (see the first two groups (blue and red) in the example of figure 1). Here, the feature vectors are sparse and only two elements are non-zero (the active user u and active item i).

1) Linear SVM: For this kind of data $\mathbf{x}$, the linear SVM model (eq. (7)) is equivalent to:

$$\hat{y}(\mathbf{x}) = w_0 + w_u + w_i \tag{10}$$

Because $x_j = 1$ if and only if $j = u$ or $j = i$. This model corresponds to one of the most basic collaborative filtering models where only the user and item biases are captured. As this model is very simple, the parameters can be estimated well even under sparsity. However, the empirical prediction quality typically is low (see figure 2).

2) Polynomial SVM: With the polynomial kernel, the SVM can capture higher-order interactions (here between users and items). In our sparse case with $m(\mathbf{x}) = 2$, the model equation for SVMs is equivalent to:

$$\hat{y}(\mathbf{x}) = w_0 + \sqrt{2}(w_u + w_i) + w_{u,u}^{(2)} + w_{i,i}^{(2)} + \sqrt{2} w_{u,i}^{(2)}$$

First of all, w_u and $w^{(2)}_{u,u}$ express the same – i.e. one can drop one of them (e.g. $w^{(2)}_{u,u}$). Now the model equation is the same as for the linear case but with an additional user-item interaction $w^{(2)}_{u,i}$. In typical collaborative filtering (CF) problems, for each interaction parameter $w^{(2)}_{u,i}$ there is at most one observation (u,i) in the training data and for cases (u',i') in the test data there are usually no observations at all in the training data. For example in figure 1 there is just one observation for the interaction (*Alice*, *Titanic*) and non for the interaction (*Alice*, *Star Trek*). That means the maximum margin solution for the interaction parameters $w^{(2)}_{u,i}$ for all test cases (u,i) are 0 (e.g. $w^{(2)}_{A,\text{ST}} = 0$). And thus the polynomial SVM can make no use of any 2-way interaction for predicting test examples; so the polynomial SVM only relies on the user and item biases and cannot provide better estimations than a linear SVM.

For SVMs, estimating higher-order interactions is not only an issue in CF but in all scenarios where the data is hugely sparse. Because for a reliable estimate of the parameter $w^{(2)}_{i,j}$ of a pairwise interaction (i,j), there must be 'enough' cases $\mathbf{x} \in D$ where $x_i \neq 0 \wedge x_j \neq 0$. As soon as either $x_i = 0$ or $x_j = 0$, the case $\mathbf{x}$ cannot be used for estimating the parameter $w^{(2)}_{i,j}$. To summarize, if the data is too sparse, i.e. there are too few or even no cases for (i,j), SVMs are likely to fail.

C. Summary

1) The dense parametrization of SVMs requires direct observations for the interactions which is often not given in sparse settings. Parameters of FMs can be estimated well even under sparsity (see section III-A3).
2) FMs can be directly learned in the primal. Non-linear SVMs are usually learned in the dual.
3) The model equation of FMs is independent of the training data. Prediction with SVMs depends on parts of the training data (the support vectors).

V. FMs vs. other Factorization Models

There is a variety of factorization models, ranging from standard models for m-ary relations over categorical variables (e.g. MF, PARAFAC) to specialized models for specific data and tasks (e.g. SVD++, PITF, FPMC). Next, we show that FMs can mimic many of these models just by using the right input data (e.g. feature vector $\mathbf{x}$).

A. Matrix and Tensor Factorization

Matrix factorization (MF) is one of the most studied factorization models (e.g. [7], [8], [2]). It factorizes a relationship between two categorical variables (e.g. U and I). The standard approach to deal with categorical variables is to define binary indicator variables for each level of U and I (e.g. see fig. 1, first (blue) and second (red) group)[4]:

$$n := |U \cup I|, \quad x_j := \delta\left(j = i \vee j = u\right) \tag{11}$$

[4]To shorten notation, we address elements in $\mathbf{x}$ (e.g. x_j) and the parameters both by numbers (e.g. $j \in \{1,\ldots,n\}$) and categorical levels (e.g. $j \in (U \cup I)$). That means we implicitly assume a bijective mapping from numbers to categorical levels.

A FM using this feature vector $\mathbf{x}$ is identical to the matrix factorization model [2] because x_j is only non-zero for u and i, so all other biases and interactions drop:

$$\hat{y}(\mathbf{x}) = w_0 + w_u + w_i + \langle \mathbf{v}_u, \mathbf{v}_i \rangle \tag{12}$$

With the same argument, one can see that for problems with more than two categorical variables, FMs includes a nested parallel factor analysis model (PARAFAC) [1] .

B. SVD++

For the task of rating prediction (i.e. regression), Koren improves the matrix factorization model to the SVD++ model [2]. A FM can mimic this model by using the following input data $\mathbf{x}$ (like in the first three groups of figure 1):

$$n := |U \cup I \cup L|, \quad x_j := \begin{cases} 1, & \text{if } j = i \vee j = u \\ \frac{1}{\sqrt{|N_u|}}, & \text{if } j \in N_u \\ 0, & \text{else} \end{cases}$$

where N_u is the set of all movies the user has ever rated[5]. A FM ($d = 2$) would behave the following using this data:

$$\hat{y}(\mathbf{x}) = \overbrace{w_0 + w_u + w_i + \langle \mathbf{v}_u, \mathbf{v}_i \rangle + \frac{1}{\sqrt{|N_u|}} \sum_{l \in N_u} \langle \mathbf{v}_i, \mathbf{v}_l \rangle}^{\text{SVD++}}$$
$$+ \frac{1}{\sqrt{|N_u|}} \sum_{l \in N_u} \left(w_l + \langle \mathbf{v}_u, \mathbf{v}_l \rangle + \frac{1}{\sqrt{|N_u|}} \sum_{l' \in N_u, l' > l} \langle \mathbf{v}_l, \mathbf{v}'_l \rangle \right)$$

where the first part is exactly the same as the SVD++ model. 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.

C. PITF for Tag Recommendation

The problem of tag prediction is defined as ranking tags for a given user and item combination. That means there are three categorical domains involved: users U, items I and tags T. In the ECML/PKDD Discovery Challenge about tag recommendation, a model based on factorizing pairwise interactions (PITF) has achieved the best score [3]. We will show how a FM can mimic this model. A factorization machine with binary indicator variables for the active user u, item i and tag t results in the following model:

$$n := |U \cup I \cup T|, \quad x_j := \delta\left(j = i \vee j = u \vee j = t\right) \tag{13}$$

$$\Rightarrow \hat{y}(\mathbf{x}) = w_0 + w_u + w_i + w_t + \langle \mathbf{v}_u, \mathbf{v}_i \rangle + \langle \mathbf{v}_u, \mathbf{v}_t \rangle + \langle \mathbf{v}_i, \mathbf{v}_t \rangle$$

As this model is used for ranking between two tags t_A, t_B within the same user/item combination (u,i) [3], both the optimization and the prediction always work on differences between scores for the cases (u,i,t_A) and (u,i,t_B). Thus

[5]To distinguish elements in N_u from elements in I, they are transformed with any bijective function $\omega : I \to L$ into a space L with $L \cap I = \emptyset$.

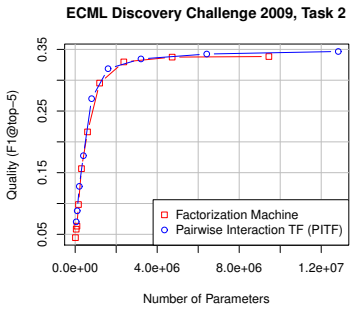

Fig. 3. Recommendation quality of a FM compared to the winning PITF model [3] of the ECML/PKDD Discovery Challenge 2009. The quality is plotted against the number of model parameters.

with optimization for pairwise ranking (like in [5], [3]), the FM model is equivalent to:

$$\hat{y}(\mathbf{x}) := w_t + \langle \mathbf{v}_u, \mathbf{v}_t \rangle + \langle \mathbf{v}_i, \mathbf{v}_t \rangle \tag{14}$$

Now the original PITF model [3] and the FM model with binary indicators (eq. (14)) are almost identical. The only difference is that (i) the FM model has a bias term w_t for t and (ii) the factorization parameters for the tags ($\mathbf{v}_t$) between the (u,t)- and (i,t)-interaction are shared for the FM model but individual for the original PITF model. Besides this theoretical analysis, figure 3 shows empirically that both models also achieve comparable prediction quality for this task.

D. Factorized Personalized Markov Chains (FPMC)

The FPMC model [4] tries to rank products in an online shop based on the last purchases (at time $t-1$) of the user u.

Again just by feature generation, a factorization machine ($d = 2$) behaves similarly:

$$n := |U \cup I \cup L|, \quad x_j := \begin{cases} 1, & \text{if } j = i \vee j = u \\ \frac{1}{|B^u_{t-1}|}, & \text{if } j \in B^u_{t-1} \\ 0, & \text{else} \end{cases} \tag{15}$$

where $B^u_t \subseteq L$ is the set ('basket') of all items a user u has purchased at time t (for details see [4]). Then:

$$\hat{y}(\mathbf{x}) = w_0 + w_u + w_i + \langle \mathbf{v}_u, \mathbf{v}_i \rangle + \frac{1}{|B^u_{t-1}|} \sum_{l \in B^u_{t-1}} \langle \mathbf{v}_i, \mathbf{v}_l \rangle$$

$$+ \frac{1}{|B^u_{t-1}|} \sum_{l \in B^u_{t-1}} \left(w_l + \langle \mathbf{v}_u, \mathbf{v}_l \rangle + \frac{1}{|B^u_{t-1}|} \sum_{l' \in B^u_{t-1}, l' > l} \langle \mathbf{v}_l, \mathbf{v}'_l \rangle \right)$$

Like for tag recommendation this model is used and optimized for ranking (here ranking items i) and thus only score differences between (u, i_A, t) and (u, i_B, t) are used in the prediction and optimization criterion [4]. Thus, all additive terms that do not depend on i vanish and the FM model equation is equivalent to:

$$\hat{y}(\mathbf{x}) = w_i + \langle \mathbf{v}_u, \mathbf{v}_i \rangle + \frac{1}{|B^u_{t-1}|} \sum_{l \in B^u_{t-1}} \langle \mathbf{v}_i, \mathbf{v}_l \rangle \tag{16}$$

Now one can see that the original FPMC model [4] and the FM model are almost identical and differ only in the additional item bias w_i and the sharing of factorization parameters of the FM model for the items in both the (u,i)- and (i,l)-interaction.

E. Summary

1) 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.
2) There are many proposals for specialized factorization models designed for a single task. We have shown that factorization machines can mimic many of the most successful factorization models (including MF, PARAFAC, SVD++, PITF, FPMC) just by feature extraction which makes FM easily applicable in practice.

VI. Conclusion and Future Work

In this paper, we have introduced factorization machines. FMs bring together the generality of SVMs with the benefits of factorization models. In contrast to SVMs, (1) FMs are able to estimate parameters under huge sparsity, (2) the model equation is linear and depends only on the model parameters and thus (3) they can be optimized directly in the primal. The expressiveness of FMs is comparable to the one of polynomial SVMs. In contrast to tensor factorization models like PARAFAC, FMs are a general predictor that can handle any real valued vector. Moreover, simply by using the right indicators in the input feature vector, FMs are identical or very similar to many of the specialized state-of-the-art models that are applicable only for a specific task, among them are biased MF, SVD++, PITF and FPMC.

References

[1] R. A. Harshman, "Foundations of the parafac procedure: models and conditions for an 'exploratory' multimodal factor analysis." *UCLA Working Papers in Phonetics*, pp. 1–84, 1970.

[2] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2008, pp. 426–434.

[3] 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*. New York, NY, USA: ACM, 2010, pp. 81–90.

[4] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *WWW '10: Proceedings of the 19th international conference on World wide web*. New York, NY, USA: ACM, 2010, pp. 811–820.

[5] T. Joachims, "Optimizing search engines using clickthrough data," in *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2002, pp. 133–142.

[6] V. N. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.

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

[8] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo," in *Proceedings of the International Conference on Machine Learning*, vol. 25, 2008.