Chapter 5
Factorization Models

In the last chapters, it was shown that a context-aware ranking $\succ$ can be expressed by a function $y : \mathcal{X} \to \mathbb{R}$ or equivalently by a tensor $Y \in \mathbb{R}^{\mathcal{X}}$ in case of finite categorical domains X_i. Estimating the full parametrized tensor Y is infeasible because (1) for real-world problems, the number of parameters (i.e. $|\mathcal{X}|$) would be too large – e.g. for the Netflix[1] problem we would need billions of parameters – and (2) even more important, that the observations are typically very sparse which results in poor estimates without any generalization capabilities.

In this chapter, we discuss how to model the tensor Y by a factorization model $\hat{Y}$. In factorization models, each variable instance x_i is expressed by a real valued vector $\mathbf{v} \in \mathbb{R}^k$ of k factors. For reconstructing the tensor $\hat{Y}$, the factors $\mathbf{v}_1, \ldots \mathbf{v}_m$ of each entry $\mathbf{x} = (x_1, \ldots, x_m) \in \mathcal{X}$ are combined. This combination is defined by the factorization model and consists usually of summations and multiplications.

We discuss linear models that are based on the Tucker decomposition which is a tensor product of factorization matrices of each mode. The structure of the Tucker decomposition is defined by its core tensor. This core tensor makes the reconstruction runtime of the Tucker decomposition exponential in the number of modes. By choosing a special core tensor, the Tucker model simplifies to PARAFAC (parallel factor analysis) which has only linear runtime. Furthermore, we will introduce the more specialized model PITF (pairwise interaction tensor factorization) that is a special case of PARAFAC. PITF models all pairwise interactions between variables explicitly. Even though the number pairwise interactions has quadratic growth in the number of modes, we show that for ranking problems the number is linear. We finish our analysis with a comparison of the runtime and expressiveness of these models.

In this whole chapter, we assume that the domain of each X_i is finite. This holds for most domains of our applications like users, items, tags, articles, web pages, words, etc. But obviously this does not hold for continuous domains like time. In chapter 8 and 9, we develop two methods for integrating time into factorization models. The first one is based on a Markov chain and the second one of modelling time variance within each factor.

[1] `http://www.netflixprize.com/`

S. Rendle: Context-Aware Ranking with Factorization Models, SCI 330, pp. 51–65.
springerlink.com

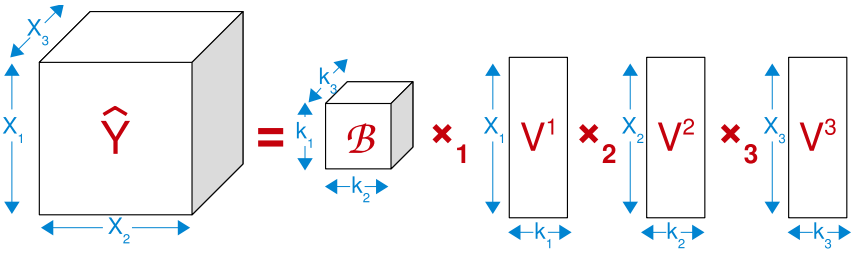

Fig. 5.1 Tucker decomposition (3-mode): The target Y is approximated by a factorization $\hat{Y}$ using a core tensor $\mathscr{B}$ and for each domain X_i one factor matrix V_i of dimensionality k_i.

5.1 Tucker Decomposition (TD)

Tucker decomposition (Tucker, 1966) factorizes a higher-order tensor into a smaller core tensor and one factor matrix for each mode. Figure 5.1 shows an example for three modes ($m = 3$).

The Tucker decomposition (TD) is defined by the following tensor product:

$$\hat{Y}^{\text{TD}} := \mathscr{B} \times_1 V^1 \ldots \times_m V^m \tag{5.1}$$

with model parameters:

$$\begin{aligned} &\mathscr{B} \in \mathbb{R}^{k_1 \times \ldots \times k_m} \\ &V^i \in \mathbb{R}^{|X_i| \times k_i}, \quad \forall i \in \{1, \ldots, m\} \end{aligned} \tag{5.2}$$

$\mathscr{B}$ is called the *core tensor*. V_i is the factor matrix for the categorical variable over the domain X_i where each row $\mathbf{v}^i_j$ describes one variable instance $x_j \in X_i$. Each entry $v^i_{j,f}$ in the matrix V^i is called a factor. k_i is the dimensionality of the factorization for X_i. That means each variable instance is expressed by k_i real numbers.

It is important to note that none of the factors are observed but they are estimated from the data. The number of factors k_i that should be used for describing a variable, is a hyperparameter that is chosen e.g. by the holdout method.

5.1.1 Model Equation

The TD corresponds to the following model equation:

$$\hat{y}^{\text{TD}}_{x_1,\ldots,x_m} := \sum_{f_1=1}^{k_1} \cdots \sum_{f_m=1}^{k_m} b_{f_1,\ldots,f_m} \prod_{i=1}^{m} v^i_{x_i,f_i} \tag{5.3}$$

As we have discussed before, the major problem in context-aware ranking tasks is the sparsity of the data. Factorization models like TD solve this problem by parametrizing variable instances instead of relation instances. The model equation

predicts relation instances by combining the individual parametrization of variable instances. This way, the model can generalize to relation instances (combinations of variable instances) that have never been observed jointly. During the learning stage, the parameters for each variable instance are optimized such that using them in the model equation (5.3) leads to an optimal prediction (in our case ranking) for the observed cases (in our case ranking pairs).

5.1.2 Gradients

Learning of the model parameters can be done by gradient descent based algorithms like BCR-LEARN (see section 4.2.2). To apply these algorithms, the partial derivatives of the model equation have to be known. The gradients of eq. (5.3) for each model parameter with respect to an instance $\mathbf{x} = (x_1, \ldots, x_m)$ are:

$$\frac{\partial \hat{y}^{\mathrm{TD}}_{x_1,\ldots,x_m}}{\partial b_{f_1,\ldots,f_m}} = \prod_{i=1}^{m} v^i_{x_i,f_i} \tag{5.4}$$

$$\frac{\partial \hat{y}^{\mathrm{TD}}_{x_1,\ldots,x_m}}{\partial v^j_{x_j,f_j}} = \sum_{f_1=1}^{k_1} \cdots \sum_{f_{j-1}=1}^{k_{j-1}} \sum_{f_{j+1}=1}^{k_{j+1}} \cdots \sum_{f_m=1}^{k_m} b_{f_1,\ldots,f_m} \prod_{i=1, i\neq j}^{m} v^i_{x_i,f_i} \tag{5.5}$$

When $\mathbf{x}$ is given, the gradient of each parameter b of the core $\mathscr{B}$ is defined because every factor of the core is always used in the model equation. In contrast to this, the gradients of the factorization matrices are only defined (non zero) for the m rows of the m variable instances in $\mathbf{x}$. All other entries of the factor matrices are not updated when $\mathbf{x}$ is fixed.

5.1.3 Complexity

Next, we analyse the complexity of TD in terms of number of parameters, number of operations in the model equation and number of operations for updating with respect to one instance $\mathbf{x}$. An overview of this analysis for all models can be found in table 5.1.

- **Number of free parameters:** The number of free parameters of eq. (5.2) is:

$$\sum_{i=1}^{m} k_i |X_i| + \prod_{i=1}^{m} k_i \tag{5.6}$$

  The first term is the number of parameters for the factor matrices and the second term is the number of core factors. To facilitate the analysis and for better comparability, we assume that all factorization dimensions are equally large, i.e. $k_1 = \ldots = k_m =: k$. Now the number of free parameters is:

$$k \sum_{i=1}^{m} |X_i| + k^m \tag{5.7}$$

As you can see, the number of parameters is exponential in m and polynomial in k.

- **Computation of Model Equation:** Predicting one entry $\mathbf{x}$ of Y with eq. (5.3) requires computing the m nested summations. In total, the number of operations is:

$$(k_1 - 1) \cdot \ldots \cdot (k_m - 1) \cdot m \tag{5.8}$$

With the assumption of equal sized factor dimensions this is:

$$m\,(k-1)^m \in O(m\,k^m) \tag{5.9}$$

Thus, prediction is exponential in the number of modes m and polynomial in the number of factors.

- **Computation of the Gradients:** Also the computation complexity of the gradient for each parameter is in the same class. There are k^m core factors that are updated; calculating each gradient requires $(m-1)$ operations. Additionally, there are $k\,m$ factors in the matrices that require each $(m-1)\,(k-1)^m$ operations. In total, this leads to:

$$(m^2 - 1)k^m \in O(m^2\,k^m) \tag{5.10}$$

Storing intermediate gradients for each factor within a mode reduces this to $O(m\,k^m)$.

According to this analysis, TD has problems to scale to problems with many modes or large factor dimensions because the number of model parameters as well as prediction and update is exponential in m and polynomial in k.

5.1.4 Two-Mode Tucker Decomposition

Next, we want to emphasise that TD for two mode problems is different from typical matrix factorization (Srebro et al, 2005) where the model equation is defined as:

$$\begin{aligned} \hat{y}^{\mathrm{MF}}_{x_1,x_2} &:= \sum_{f}^{k} v^1_{x_1,f} \cdot v^2_{x_2,f} \\ &= \left\langle \mathbf{v}^1_{x_1}, \mathbf{v}^2_{x_2} \right\rangle \end{aligned} \tag{5.11}$$

The differences between TD and MF can be seen by writing the TD model equation explicitly for $m = 2$:

$$\hat{y}^{\mathrm{TD}}_{x_1,x_2} := \sum_{f_1=1}^{k_1} \sum_{f_2=1}^{k_2} b_{f_1,f_2} \cdot v^1_{x_1,f_1} \cdot v^2_{x_2,f_2} \tag{5.12}$$

Now, the differences are obvious: (1) 2-mode TD contains two nested sums and a full parametrized core matrix. By setting the core matrix to a diagonal matrix, the usual two-mode SVD-scheme $U\Sigma V$ is obtained, where $U = V^1$, $B = \Sigma$ and $V = V^2$.

5.1.5 Higher-Order SVD

Higher-order singular value decomposition (HOSVD) uses the same model equation as TD but HOSVD implies also the choice of the optimization criterion that is mean square error. Thus, HOSVD reconstructs a fully observed tensor Y for minimal square loss. In our case with many missing values, the non-observed values have to be imputed (e.g. by 0) which results in very huge sparsity (see section 3.1.4). We will compare our approaches empirically to HOSVD and SVD in chapter 6 and 7.

5.2 Parallel Factor Analysis (PARAFAC)

The PARAFAC model (Harshman, 1970) is a special case of the Tucker decomposition that assumes a diagonal and constant core tensor. An example for three modes ($m = 3$) is shown in figure 5.2.

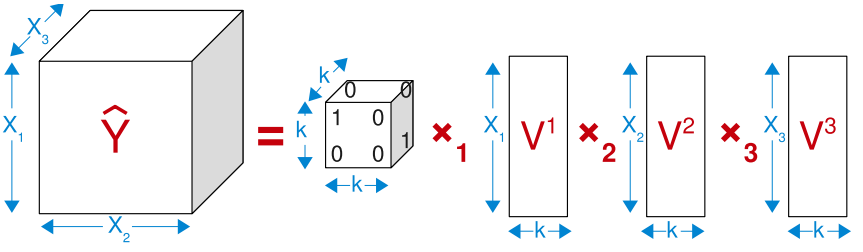

Fig. 5.2 PARAFAC: The model has a diagonal core tensor and the factorization dimensionality is equal for all modes.

PARAFAC uses the same decomposition (eq. (5.1)) and parametrization (eq. (5.2)) as TD but in PARAFAC, the dimensionalities of the factor matrices are identical:

$$\forall i \in \{1,\ldots,m\}: \quad k_i := k \tag{5.13}$$

and the core tensor is diagonal:

$$b_{f_1,\ldots,f_m} := \begin{cases} 1, & \text{if } f_1 = \ldots = f_m \\ 0, & \text{else} \end{cases} \tag{5.14}$$

This leads to model equation where only the factors within the same dimension are linked. This can be seen as a hard-wired core tensor that predefines the interactions between factors whereas in TD this core structure is flexible.

5.2.1 Model Equation

In total, the model equation for PARAFAC is:

$$\hat{y}^{\text{PARAFAC}}_{x_1,\ldots,x_m} := \sum_{f=1}^{k} \prod_{i=1}^{m} v^i_{x_i,f} \tag{5.15}$$

Thus, in PARAFAC all factor values within a given dimension f are multiplied. These products are built over all dimensions and summed up. Even though, the factor matrices itself are not restricted, the model equation is simplified a lot by keeping the core tensor diagonal and constant. Thus, in PARAFAC only the factor matrices have to be learned.

5.2.2 Gradients

For optimizing the parameters of PARAFAC with a gradient descent based algorithm, we state the derivates. The gradients of eq. (5.15) for each model parameter with respect to an instance $\mathbf{x} = (x_1, \ldots, x_m)$ are:

$$\frac{\partial \hat{y}^{\text{PARAFAC}}_{x_1,\ldots,x_m}}{\partial v^j_{x_j,f}} = \prod_{i=1, i\neq j}^{m} v^i_{x_i,f_i} \tag{5.16}$$

5.2.3 Complexity

Next, we will show the influence of fixing the core on the number of free parameters and the computation of prediction and gradients.

- **Number of free parameters:** In general, PARAFAC uses the same decomposition as TD, but by fixing the core tensor to a constant tensor eq. (5.14), the number of free parameters is reduced to:

$$k \sum_{i=1}^{m} |X_i| \tag{5.17}$$

  Now, the number of parameters is linear in k and the sizes of the domains.
- **Computation of Model Equation:** Predicting one entry $\mathbf{x}$ of Y with PARAFAC (eq. (5.15)) requires summation over k products of size m. In total, the number of operations is:

$$(k-1)(m-1) \in O(km) \tag{5.18}$$

  Thus, the computation of one element is both linear in k and m.
- **Computation of the Gradients:** For updating with respect to a fixed entry $\mathbf{x}$, all factors of the related variable instances are updated. In total, the number of

updated factors is $k \cdot m$. And for each update with eq. (5.16), the number of operations to calculate the gradient is $m-2$. This results in a total number of update operations of:

$$k \cdot m \cdot (m-2) \in O(k m^2) \tag{5.19}$$

By precalculating $\alpha_f := \prod_{i=1}^{m} v^i_{x_i,f}$ for each factor, the calculation of the gradient with respect to $v^i_{x_i,f}$ can be performed in $O(1)$ with $\frac{\alpha_f}{v^i_{x_i,f}}$. Thus the total runtime for the update drops to $O(k m)$.

This analysis shows that PARAFAC scales much better than TD with respect to m and k. Even though PARAFAC and TD have similar model equations, by setting the core tensor constant and diagonal the complexity drops largely because the interactions between factors are reduced to interactions within the same dimension.

5.2.4 Two-Mode PARAFAC

In the case of two modes ($m = 2$), PARAFAC is identical to the popular Matrix Factorization model (eq. (5.11)):

$$\hat{y}^{\text{PARAFAC}}_{x_1,x_2} = \sum_{f=1}^{k} \prod_{i=1}^{2} v^i_{x_i,f} = \sum_{f=1}^{k} v^1_{x_1,f} \cdot v^2_{x_2,f} = \langle \mathbf{v}^1_{x_1}, \mathbf{v}^2_{x_2} \rangle = \hat{y}^{\text{MF}}_{x_1,x_2} \tag{5.20}$$

That means that PARAFAC is a higher-order counter part of matrix factorization. In the following, we will present another model which is also a higher-order counter part of MF.

5.3 Pairwise Interaction Tensor Factorization (PITF)

Instead of modelling an m-ary relation directly with one m-ary product like PARAFAC, PITF models many pairwise relations. The idea is to model one interaction explicitly for each variable pair (Rendle and Schmidt-Thieme, 2010). In general, there are $\frac{m(m-1)}{2} \in O(m^2)$ distinct pairs of variables. For higher order problems (i.e. large m), modelling all pairwise interactions independently is not feasible. But, we will show that for learning ranking problems and optimizing with BCR, both the prediction and learning is invariant to many of these interactions and thus they can be dropped. In total, this will lead to only $(m-1) \in O(m)$ pairwise interactions that are relevant for ranking. Furthermore, we will show that PITF is a special case of PARAFAC and thus also of TD.

5.3.1 Model Equation

The general model equation for PITF without pruning is:

$$\hat{y}^{\text{G-PITF}}_{x_1,\ldots,x_m} := \sum_{i=1}^{m}\sum_{j=i+1}^{m} \langle \mathbf{v}^{i,j}_{x_i}, \mathbf{v}^{j,i}_{x_j} \rangle = \sum_{i=1}^{m}\sum_{j=i+1}^{m}\sum_{f=1}^{k_{i,j}} v^{i,j}_{x_i,f} \cdot v^{j,i}_{x_j,f} \tag{5.21}$$

with model parameters:

$$V^{i,j} \in \mathbb{R}^{|X_i| \times k_{i,j}},\ V^{j,i} \in \mathbb{R}^{|X_j| \times k_{i,j}}, \quad \forall i,j \in \{1,\ldots,m\},\ i > j \tag{5.22}$$

That means, there are $\frac{m(m-1)}{2}$ factorization pairs.

5.3.1.1 Relationship to PARAFAC and TD

Next, we will discuss the relationship between PITF and both PARAFAC and TD. Figure 5.3 shows how a general 3-mode PITF model can be expressed by the Tucker Decomposition.

Lemma 5.1 (PITF is a special case of PARAFAC). *Every PITF model can be expressed by a PARAFAC model with* $k = \sum_{i=1}^{m}\sum_{j=i+1}^{m} k_{i,j}$ *dimensions.*

Proof. Assume, an arbitrary PITF model is given and let $k_{i,j}$ be its pairwise factor dimensionality. The corresponding PARAFAC model of order $k = \sum_{i=1}^{m}\sum_{j=i+1}^{m} k_{i,j}$ can be generated from the PITF model by setting parts of the feature matrices to constant 1. To enhance readability of this proof, we denote parameters of the PITF model by U and the parameters of the PARAFAC model by V.

Let Z_p be the set of all ordered pairs:

$$Z_p := \{(i,j) \,|\, i,j \in \{1,\ldots,m\}, i > j\} \tag{5.23}$$

It is known that $|Z_p| = \frac{m(m-1)}{2}$. Now let Z be the set of all ordered index pairs with indices over their factorization dimensionality in PITF:

$$Z := \{(i,j,f) \,|\, (i,j) \in Z_p, f \in \{1,\ldots,k_{i,j}\}\} \tag{5.24}$$

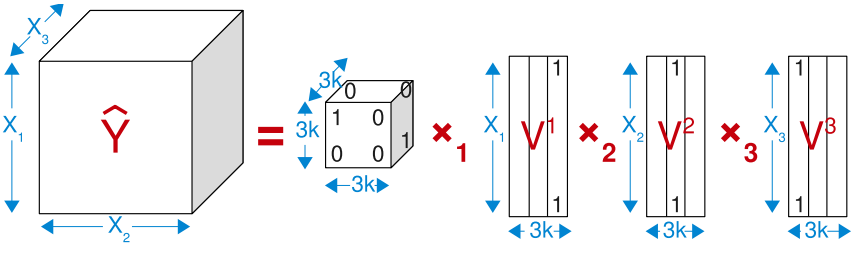

Fig. 5.3 General PITF: Like in PARAFAC, the core tensor is diagonal, but in G-PITF one third of each factor matrix V_i is fixed to constant 1. This results in three pairwise interactions: (x_1,x_2), (x_1,x_3) and (x_2,x_3).

The cardinality of Z is the size of the corresponding PARAFAC model:

$$|Z| = k = \sum_{i=1}^{m} \sum_{j=i+1}^{m} k_{i,j} \tag{5.25}$$

Let ϕ be a bijective mapping from Z to $\{1,\ldots,|Z|\}$. With this, we can rewrite the PARAFAC model:

$$\hat{y}^{\text{PARAFAC}}_{x_1,\ldots,x_m} = \sum_{f=1}^{k} \prod_{i=1}^{m} v^{i}_{x_i,f} = \sum_{(i,j,f)\in Z} \prod_{l=1}^{m} v^{l}_{x_l,\phi(i,j,f)} \tag{5.26}$$

Now, we set some of the parameters of the PARAFAC model to constants:

$$v^{l}_{x_l,\phi(i,j,f)} = \begin{cases} u^{i,j}_{x_i,f} & \text{if } l = i \\ u^{j,i}_{x_j,f} & \text{if } l = j \\ 1, & \text{else} \end{cases} \tag{5.27}$$

Subsituted in eq. (5.26) this leads to:

$$\begin{aligned} \hat{y}^{\text{PARAFAC}}_{x_1,\ldots,x_m} &= \sum_{(i,j,f)\in Z} \prod_{l=1}^{m} v^{l}_{x_l,\phi(i,j,f)} = \sum_{(i,j,f)\in Z} u^{i,j}_{x_i,f}\, u^{j,i}_{x_j,f} = \sum_{i=1}^{m} \sum_{j=i+1}^{m} \sum_{f=1}^{k_{i,j}} u^{i,j}_{x_i,f}\, u^{j,i}_{x_j,f} \\ &= \sum_{i=1}^{m} \sum_{j=i+1}^{m} \langle \mathbf{u}^{i,j}_{x_i}, \mathbf{u}^{j,i}_{x_j} \rangle = \hat{y}^{\text{G-PITF}}_{x_1,\ldots,x_m} \end{aligned} \tag{5.28}$$

Note that this proofs shows that every PITF model can be expressed by a PARAFAC model. It is important to note that the contradiction does not hold, i.e. a general PARAFAC model of mode $m \neq 2$ can not be represented by a PITF model.

5.3.1.2 PITF for Ranking

The general PITF model eq. (5.21) requires a large number of parameters, as the number of variable pairs growths quadratic in m. Next, we show that when PITF is used for ranking with respect to the variable domain X_m, many pairwise interactions can be dropped without loosing expressiveness.

The model equation for ranking PITF is:

$$\hat{y}^{\text{PITF}}_{x_1,\ldots,x_m} := \sum_{i=1}^{m-1} \langle \mathbf{v}^{i,m}_{x_i}, \mathbf{v}^{m,i}_{x_m} \rangle \tag{5.29}$$

with model parameters:

$$V^{i,m} \in \mathbb{R}^{|X_i|\times k_{i,m}},\ V^{m,i} \in \mathbb{R}^{|X_m|\times k_{i,m}}, \quad \forall i \in \{1,\ldots,m-1\} \tag{5.30}$$

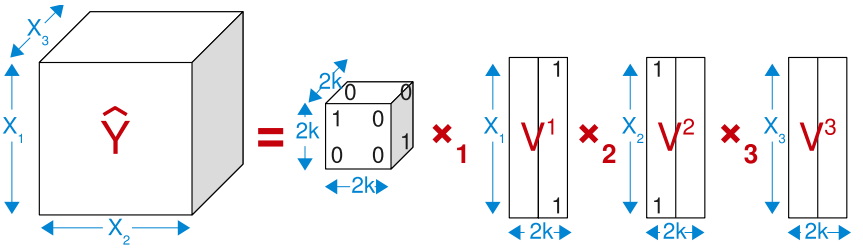

Fig. 5.4 PITF for Ranking: From the general PITF, the interaction (x_1, x_2) can be removed as it has no influence on ranking with respect to x_3. This holds only for predicting rankings and optimizing the parameters for ranking (like with BCR-OPT).

Before showing that this model is equivalent to the general PITF model for the task of ranking, we first note the general expressiveness and the relationship to PARAFAC and TD. (i) It is clear, that every ranking PITF model is also a general PITF model. This can be seen by setting $k_{i,j} = 0, \forall i \neq m, j \neq m$. And (ii) thus also every PITF is a special case of PARAFAC and TD. An example for three modes is shown in figure 5.4.

Lemma 5.2 (Invariance of Pairs). *The G-PITF (eq. (5.21)) and PITF (eq. (5.29)) model are invariant under ranking (eq. (3.34)) and learning with respect to BCR (eq. (4.11)).*

Proof. G-PITF can be rewritten as:

$$\hat{y}^{\text{G-PITF}}_{x_1,\ldots,x_m} = \hat{y}^{\text{PITF}}_{x_1,\ldots,x_m} + z_{x_1,\ldots,x_{m-1}} \tag{5.31}$$

with

$$z_{x_1,\ldots,x_{m-1}} := \sum_{i=1}^{m-1} \sum_{j=i+1}^{m-1} \langle \mathbf{v}^{i,j}_{x_i}, \mathbf{v}^{j,i}_{x_j} \rangle \tag{5.32}$$

Now G-PITF consists of a term PITF that depends on x_m and a term $z_{\mathbf{c}}$ that depends not on x_m but only on the context $\mathbf{c} = (x_1, \ldots, x_{m-1})$. The equivalence of PITF and G-PITF with respect to ranking and learning with BCR is proofed by the following stronger lemma.

Lemma 5.3 (Invariance of Additive Terms). $\hat{y}_{x_1,\ldots,x_m}$ *and* $\hat{y}'_{x_1,\ldots,x_m} := \hat{y}_{x_1,\ldots,x_m} + z_{\mathbf{c}}$ *are invariant to ranking and learning with BCR.*

Proof. First, we show the invariance to ranking. Let $\succ$ be the ranking induced by $\hat{y}$ and $\succ'$ be the ranking induced by $\hat{y}'$:

$$
\begin{aligned}
x_i \succ_{\mathbf{c}} x_j &\overset{eq.(3.34)}{\Leftrightarrow} \hat{y}_{\mathbf{c},x_i} > \hat{y}_{\mathbf{c},x_j} \\
&\Leftrightarrow \hat{y}_{\mathbf{c},x_i} + z_{\mathbf{c}} > \hat{y}_{\mathbf{c},x_j} + z_{\mathbf{c}} \\
&\Leftrightarrow \hat{y'}_{\mathbf{c},x_i} > \hat{y'}_{\mathbf{c},x_j} \overset{eq.(3.34)}{\Leftrightarrow} x_i \succ'_{\mathbf{c}} x_j
\end{aligned}
$$

Secondly, we show the invariance for learning with BCR. Recall the definition of BCR-OPT:

$$
\begin{aligned}
&\underset{\Theta}{\operatorname{argmax}}\, \text{BCR-OPT} \\
=&\underset{\Theta}{\operatorname{argmax}} \sum_{(\mathbf{c},x_i,x_j)\in\mathscr{C}\times X_m^2} d_s(\mathbf{c},x_i,x_j)\, \ln\sigma(\hat{y}(\mathbf{c},x_i)-\hat{y}(\mathbf{c},x_j)) - \sum_{\theta\in\Theta}\frac{1}{2}\lambda_\theta\,\theta^2
\end{aligned}
$$

The model $\hat{Y}$ is only used as the difference of two variable instances $x_i, x_j \in X_m$. Thus:

$$
\begin{aligned}
\hat{y}'(\mathbf{c},x_i) - \hat{y}'(\mathbf{c},x_j) &= \hat{y}'_{\mathbf{c},x_i} - \hat{y}'_{\mathbf{c},x_j} = \hat{y}_{\mathbf{c},x_i} + z_{\mathbf{c}} - \hat{y}_{\mathbf{c},x_j} - z_{\mathbf{c}} \\
&= \hat{y}_{\mathbf{c},x_i} - \hat{y}_{\mathbf{c},x_j} = \hat{y}(\mathbf{c},x_i) - \hat{y}(\mathbf{c},x_j)
\end{aligned}
$$

So the additional term $z_{\mathbf{c}}$ vanishes completely in the optimization. That means it is never used in the optimization function and thus it is also not updated with gradient descent learning. In total, as the term $z_{\mathbf{c}}$ is not regarded, the models $\hat{Y}$ and $\hat{Y}'$ are invariant.

Note that this only holds for optimization with a ranking criterion like BCR. For other optimization approaches like element-wise losses (e.g. LSE) the invariance does not hold and thus the interactions can not be dropped. The reason is, that these approaches try to estimate an absolute value and this value (largely) depends on the context.

5.3.2 Gradients

For learning a PITF model with gradient descent, the partial derivatives of eq. (5.29) for each model parameter with respect to an instance $\mathbf{x} = (x_1, \ldots, x_m)$ are:

$$
\frac{\partial \hat{y}^{\text{PITF}}_{x_1,\ldots,x_m}}{\partial v^{i,j}_{x_i,f}} = v^{j,i}_{x_j,f}, \quad \frac{\partial \hat{y}^{\text{PITF}}_{x_1,\ldots,x_m}}{\partial v^{j,i}_{x_j,f}} = v^{i,j}_{x_i,f} \tag{5.33}
$$

where for general PITF:

$$
i,j \in \{1,\ldots,m\}, \quad i<j \tag{5.34}
$$

and for ranking PITF:

$$
i \in \{1,\ldots,m-1\}, \quad j=m \tag{5.35}
$$

5.3.3 Complexity

Next, we discuss the complexity of the ranking PITF model in terms of number of free parameters, and number of operations for prediction and update:

- **Number of free parameters:** In ranking PITF, there is one factor matrix for each variable but for the variable to rank, there are $m-1$ independent factor matrices. In total, the number of free parameters is:

$$\sum_{i=1}^{m-1} k_{i,m} \left(|X_i| + |X_m|\right) \tag{5.36}$$

  With the assumption that all factorization dimensionalities are equal (i.e. $\forall i \in \{1,\ldots,m-1\} : k_{i,m} \stackrel{!}{=} k$), this simplifies to:

$$k \sum_{i=1}^{m} |X_i| + k\,(m-2)\,|X_m| \tag{5.37}$$

  That means like for PARAFAC, the number of free parameters for PITF is linear in k and m. But there are $k\,(m-2)\,|X_m|$ more parameters in the PITF model than in PARAFAC, when the same k is chosen.
- **Computation of Model Equation:** Predicting one entry $\mathbf{x}$ of Y with PARAFAC (eq. (5.15)) corresponds to $(m-1)$ summations over a scalar product of size k. Thus the number of operations is:

$$(m-1)\,(2k-1) \in O(km) \tag{5.38}$$

- **Computation of the Gradients:** Given a relation instance $\mathbf{x}$, all factors of the related variable instances are updated. That means, there are $km + k\,(m-2)$ parameters to update. The gradient in eq. (5.33) requires no computation time, so the number of basic operations is:

$$2k\,(m-1) \in O(km) \tag{5.39}$$

  Again, this is linear in both k and m.

5.3.4 Two-Mode PITF

Again for two-mode scenarios, general PITF and PITF for ranking is identical to matrix factorization. The reason is, that for two modes, there is only one interaction and thus:

$$\hat{y}^{\text{PITF}}_{x_1,x_2} := \langle \mathbf{v}^{1,2}_{x_1}, \mathbf{v}^{2,1}_{x_2} \rangle = \hat{y}^{\text{MF}}_{x_1,x_2} \tag{5.40}$$

Furthermore this shows that PITF and PARAFAC have the same model equation for two-mode problems.

Table 5.1 Model complexity with respect to number of free parameters and computation runtime for prediction and gradients.

Model	Free Parameters	Prediction	Gradient
TD	$k\sum_{i=1}^{m}\lvert X_i\rvert + k^m$	mk^m	mk^m
PARAFAC	$k\sum_{i=1}^{m}\lvert X_i\rvert$	mk	mk
PITF	$k\sum_{i=1}^{m}\lvert X_i\rvert + k\,(m-2)\,\lvert X_m\rvert$	mk	mk

5.4 Expressiveness

Now, we want to summarize the expressiveness of the approaches. We use $\mathscr{M}$ to denote the model classes and we write $\mathscr{M}^{\text{TD}}$ for Tucker Decomposition, $\mathscr{M}^{\text{PARAFAC}}$ for PARAFAC and $\mathscr{M}^{\text{PITF}}$ for PITF. Let $\mathscr{M}(m)$ denote the model class for a specific number of modes.

Two-Mode Problems

We have seen, that $\mathscr{M}^{\text{PARAFAC}}(2) = \mathscr{M}^{\text{PITF}}(2)$ and also that they are equivalent to usual matrix factorization $\mathscr{M}^{\text{MF}}$. Furthermore, we have shown, that $\mathscr{M}^{\text{TD}}(2) \subset \mathscr{M}^{\text{MF}}$ because the core matrix of TD allows all possible interactions between factors of different dimensions. In total, we have:

$$\mathscr{M}^{\text{TD}}(2) \supset \mathscr{M}^{\text{PARAFAC}}(2) = \mathscr{M}^{\text{PITF}}(2) = \mathscr{M}^{\text{MF}} \tag{5.41}$$

Higher-order Problems ($m \geq 3$)

The equivalence of PITF and PARAFAC only holds for $m = 2$. For higher dimensions, PITF cannot express PARAFAC as it can only model two way interactions (multiplications) directly. But we have seen, that every PITF model can be formulated as a PARAFAC model. And furthermore, TD subsumes PARAFAC for any number of modes ($m \geq 2$). To summarize, the expressiveness is:

$$\mathscr{M}^{\text{TD}}(m) \supset \mathscr{M}^{\text{PARAFAC}}(m) \supset \mathscr{M}^{\text{PITF}}(m), \quad \forall m \geq 3 \tag{5.42}$$

Discussion

At first glance, one might think that reducing the expressiveness helps only in terms of runtime. E.g. PARAFAC has a better prediction and update complexity than TD (see table 5.1). If this would be the case, PITF models would not make any sense as PARAFAC subsumes them and both have the same computational complexity. But actually, choosing a model with less expressiveness does not have to lead to worse prediction quality — i.e. that quality is traded in for e.g. runtime. We will

show this in detail for tag recommenders (chapter 7). The reason is that e.g. PITF explicitly models a structure that might be hard to find under sparsity for the TD and PARAFAC approach. Especially, regularization approaches like ridge regression (see eq. (4.11)) which usually assume that the model parameters are normally distributed with mean zero $\theta \sim \mathcal{N}\left(0, \frac{1}{2\lambda_\theta}\right)$ might fail to learn the structure modeled explicitly. Thus, if a model structure is known a priori, it might be better to model it explicitly than trying to learn it. This is especially important for our problem settings with large sparsity.

5.5 Computational Aspects

Finally, the costs of factorization models are discussed from a practical point of view.

Memory usage

A huge advantage of factorization models is that they store for each variable instance (e.g. each user) only a small number of factors (k). Such a small factor vector is the only information needed to be stored (e.g. for a customer or a product) to make predictions. Other information like the historical events (S) or user/ item attributes (like name, brand) are not necessary for predictions. Clearly, storing a small number of factors for each variable instance is feasible for any real world system – e.g. online shops typically have to store already much more information per customer or product (like name, age, address). Furthermore, factorization models allow to control the trade-off between quality and memory consumption by increasing or decreasing the number of latent factors (k).

Runtime

For the linear models (PARAFAC and PITF), prediction is fast as it only depends linearly on the precomputed factors. Again, the prediction is independent of any other information like the historical events (S) or other information about the variable instances (e.g. the name of customer, the genre of a book). This is in contrast to other models like kNN where the prediction depends on the historical events.

On the other hand, the fast prediction of factorization models comes to the prize of training the model. But in an application, training is usually done offline. Furthermore, PARAFAC and PITF are easily parallelizable because their model equation of two instances $\mathbf{x}$ and $\mathbf{x}'$ does not share any parameters if $x_i \neq x'_i, \forall i \in \{1, \ldots, m\}$ – this holds for most instances when the domains are large. Additionally, the models can also be trained with online-updates. This is important in practice because usually a model is trained offline, then it is deployed and updates should be performed online. We have provided such a online-update algorithm for regularized least-square matrix factorization (Rendle and Schmidt-Thieme, 2008) – a similar

algorithm can be used for BCR learning of PARAFAC or PITF. In the evaluation of (Rendle and Schmidt-Thieme, 2008), online updating the factors of a user or item on the large Netflix dataset ($|S| \approx 100,000,000$) costs between 0ms and 15ms (depending on the number of historical events for this user/ item). Moreover, it was shown that for variable instances with a large history, online-updating after each event is not necessary.

In all this shows that factorization models are highly applicable in practice because their memory storage is small, the predictions are fast and learning can be done by online-updating the factors in real-time.

References

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

Rendle, S., Schmidt-Thieme, L.: Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In: RecSys 2008: Proceedings of the 2008 ACM Conference on Recommender Systems, pp. 251–258. ACM, New York (2008)

Rendle, S., Schmidt-Thieme, L.: Pairwise interaction tensor factorization for personalized tag recommendation. In: WSDM 2010: Proceedings of the third ACM International Conference on Web search and Data Mining, pp. 81–90. ACM, New York (2010)

Srebro, N., Rennie, J.D.M., Jaakola, T.S.: Maximum-margin matrix factorization. In: Advances in Neural Information Processing Systems, vol. 17, pp. 1329–1336. MIT Press, Cambridge (2005)

Tucker, L.: Some mathematical notes on three-mode factor analysis. Psychometrika 31, 279–311 (1966)