Pairwise Interaction Tensor Factorization for Personalized Tag Recommendation

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

Lars Schmidt-Thieme
Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany
schmidt-thieme@ismll.uni-hildesheim.de

ABSTRACT

Tagging plays an important role in many recent websites. Recommender systems can help to suggest a user the tags he might want to use for tagging a specific item. Factorization models based on the Tucker Decomposition (TD) model have been shown to provide high quality tag recommendations outperforming other approaches like PageRank, FolkRank, collaborative filtering, etc. The problem with TD models is the cubic core tensor resulting in a cubic runtime in the factorization dimension for prediction and learning.

In this paper, we present the factorization model PITF (Pairwise Interaction Tensor Factorization) which is a special case of the TD model with linear runtime both for learning and prediction. PITF explicitly models the pairwise interactions between users, items and tags. The model is learned with an adaption of the Bayesian personalized ranking (BPR) criterion which originally has been introduced for item recommendation. Empirically, we show on real world datasets that this model outperforms TD largely in runtime and even can achieve better prediction quality. Besides our lab experiments, PITF has also won the ECML/PKDD Discovery Challenge 2009 for graph-based tag recommendation.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Parameter learning*

General Terms

Algorithms, Experimentation, Measurement, Performance

Keywords

Tag recommendation, Tensor factorization, Personalization, Recommender systems

*Steffen Rendle is currently on leave from the Machine Learning Lab, University of Hildesheim, Germany.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
WSDM'10, February 4–6, 2010, New York City, New York, USA.
Copyright 2010 ACM 978-1-60558-889-6/10/02 ...$10.00.

1. INTRODUCTION

Tagging is an important feature of the Web 2.0. It allows the user to annotate items/ resources like songs, pictures, bookmarks, etc. with keywords. Tagging helps the user to organize his items and facilitate e.g. browsing and searching. Tag recommenders assist the tagging process of a user by suggesting him a set of tags that he is likely to use for an item. Personalized tag recommenders take the user's tagging behaviour in the past into account when they recommend tags. That means each user is recommended a personalized list of tags – i.e. the suggested list of tags depends both on the user and the item. Personalization makes sense as people tend to use different tags for tagging the same item. This can be seen in systems like Last.fm that have a non-personalized tag recommender but still the people use different tags. In [18] an empirical example was shown where recent personalized tag recommenders outperform even the theoretical upper-bound for any non-personalized tag recommender.

This work builds on the recent personalized tag recommender models using factorization models. These models like Higher-Order-Singular-Value-Decomposition (HOSVD) [22] and Ranking Tensor Factorization (RTF) [18] are based on the Tucker Decomposition (TD) model. RTF has shown to result in very good prediction quality. The drawback of using full TD is that the model equation is cubic in the factorization dimension. That makes TD models using a high factorization dimension unfeasible for midsized and large datasets. In this paper, we present a new factorization model that explicitly models the pairwise interactions between users, items and tags. The advantage of this model is that the complexity of the model equation is linear in the number of factorization dimensions which makes it feasible for high dimensions. In statistics, another approach for tensor factorization with a model equation of linear complexity is the canonical decomposition (CD) [1] – aka parallel factor analysis (PARAFAC) [2]. We will show that our model is a special case of both CD and TD. Our experimental results also indicate that our pairwise interaction model clearly outperforms the CD model in prediction quality and slightly in runtime. Furthermore for learning tag recommender models in general, we adapt the Bayessian Personalized Ranking optimization criterion (BPR-Opt) [17] from item recommendation to tag recommendation.

In all, our contributions are as follows:

1. We extend the Bayessian Personalized Ranking optimization criterion (BPR-Opt) [17] to the task of

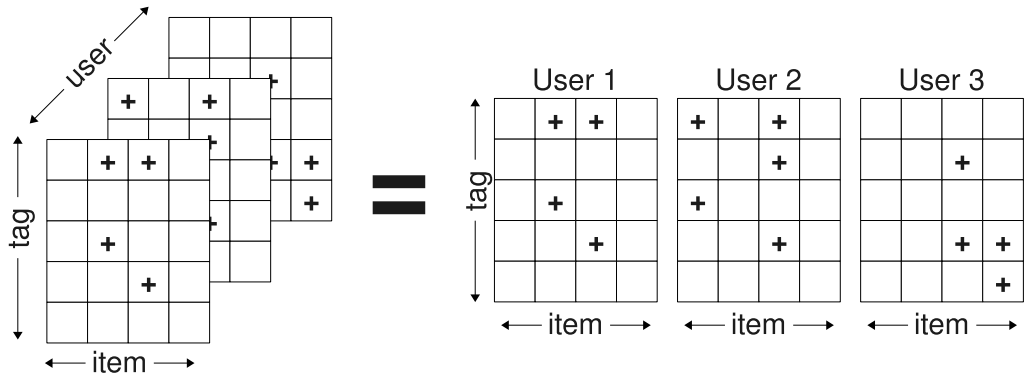


Figure 1: The observed data in a tagging system forms a ternary relation S between users U, items I and tags T. On the right side, the cube's slices per user are placed next to each other. Note that only positive observations are made; there are no explicit observations of negative tagging events.

tag recommendation and provide a learning algorithm based on stochastic gradient descent with bootstrap sampling. This optimization criterion and learning algorithm is generic and not limited to factorization models like TD.

2. We provide the factorization model PITF with a linear prediction/ reconstruction runtime. We show the relationship to the general Tucker Decomposition (TD) model and the canonical decomposition (CD; aka PARAFAC).

3. Our experiments indicate that our method BPR-PITF outperfoms the best quality method RTF-TD largely in runtime as the runtime drops from $O(k^3)$ to $O(k)$ — where k is the factorization dimension. Moreover, the quality of BPR-PITF is comparable to RTF-TD on the Bibsonomy dataset and even outperforms RTF-TD on the larger Last.fm dataset.

2. RELATED WORK

2.1 Personalized Tag Recommender

Personalized tag recommendation is a recent topic in recommender systems. FolkRank, an adaption of PageRank, was introduced by Hotho et al. [5]. FolkRank generates high quality recommendations [8] outperforming several baselines like most-popular models and collaborative filtering [7]. Recently, factorization models based on Tucker Decomposition (TD) have been introduced to tag recommendation. In [22] a Higher-Order-Singular-Value-Decomposition (HOSVD) is used – which corresponds to a TD model optimized for square-loss where all not observed values are learned as 0s. In [18] a better learning approach for TD models has been introduced, that optimizes the model parameters for the ranking statistic AUC (area under the ROC-curve). The optimization of this model is related to our proposed BPR optimization for tag recommendation because both optimize over pairs of ranking constraints. But in contrast to the AUC optimization in [18], we optimize for pair classification. A discussion of the relationship of AUC optimization and the BPR pair classification can be found in [17] which is also the basis of the BPR framework that we adapt for tag recommendation.

2.2 Non-personalized Tag Recommender

There is also much work (e.g. [3, 21]) on non-personalized tag recommenders – i.e. for a certain item they recommend all users the same tags. As discussed in the introduction, we think that personalization is important as users tend to use different tags even when they get the same suggestions. Besides this in [18] it was empirically shown for our scenarios that methods like Folkrank and RTF outperform the theoretical upper bound for any non-personalized tag recommender.

2.3 Tensor Factorization Models

Factorization models for tensors are studied in several fields for many years. A general model is the Tucker decomposition [23] on which the tag recommenders in [22, 18] are based. A special case of Tucker decomposition is the canonical decomposition (CD) [1] also known as the parallel factor analysis (PARAFAC) [2]. We discuss both TD and CD/PARAFAC in section 5 and show the relation to our factorization model. A popular approach for learning TD models is HOSVD [12]. In [18] it has been shown that for tag recommendation HOSVD results in low prediction quality and that other optimization criteria achieve better recommendations. For the related task of item recommendation, there is a detailed comparison in [17] comparing BPR-optimization to regularized sparse least-square matrix factorization like in [6, 16]

2.4 Pairwise Interaction Model

We have introduced our method for task 2 of the ECML/ PKDD Discovery Challenge [19] where it scored first place outperforming all other approaches in prediction quality. An overview of our approach for the challenge has been presented in the workshop proceedings [19]. This paper differs from [19] by providing a more detailed and general overview: (1) We show the relations to other approaches like the TD based approaches RTF and HOSVD as well as the CD model. (2) We empirically compare our approach to state-of-the-art methods on other tag recommendation datasets. (3) This paper also introduces the related CD model to tag recommendation and shows its prediction quality.

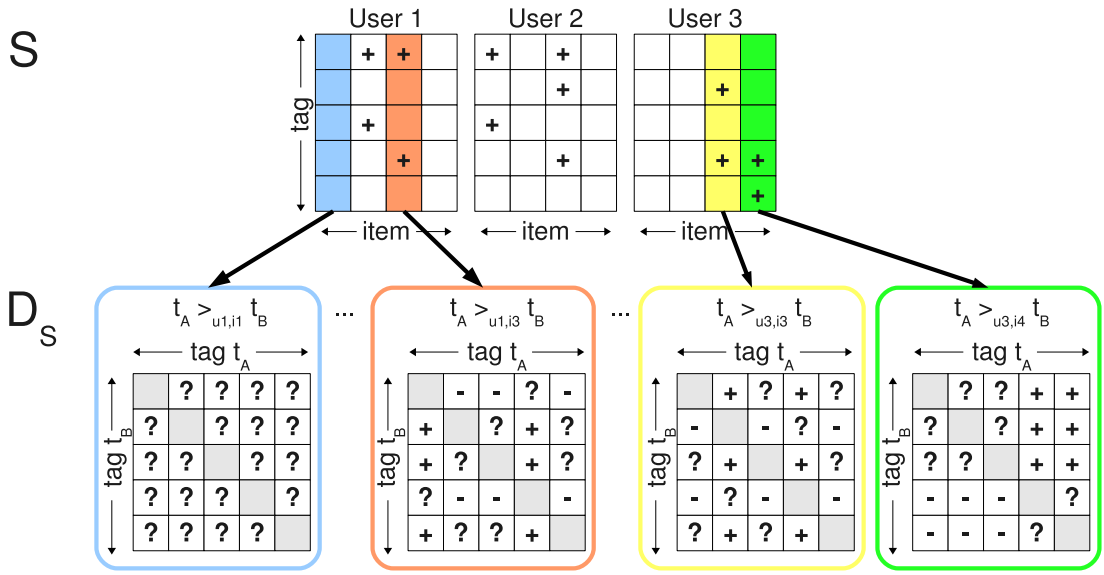


Figure 2: From the observed data S, pairwise preferences D_S of tags can be inferred per post (user/ item combination). On the bottom there are examples for four posts: (u_1, i_1) (blue), (u_1, i_3) (red), (u_3, i_3) (yellow) and (u_3, i_4) (green). E.g. for post (u_1, i_3), the following positive constraints can be inferred: $t_1 >_{u_1,i_3} t_2$, $t_1 >_{u_1,i_3} t_3$, $t_1 >_{u_1,i_3} t_5$, $t_4 >_{u_1,i_3} t_2$, $t_4 >_{u_1,i_3} t_3$, $t_4 >_{u_1,i_3} t_5$. For posts without any observed tag (like (u_1, i_1)), no constraints can be inferred.

3. PERSONALIZED TAG RECOMMENDATION

Personalized tag recommendation is the task of recommending a list of tags to a user for annotating (e.g. describing) an item. An example is a music website where a listener (user) wants to tag a song (item) and the system recommends him a list of keywords that the listener might want to use for this song. For inferring the recommendation list, a personalized tag recommender can use the historical data of the system, i.e. the tagging behaviour of the past. E.g. the recommender can make use of the tags that this user has given to other (similar) items in the past – or more general of similar tags that similar users haven given to similar items.

3.1 Formalization

For the formalization of personalized tag recommendation, we use the notation of [18]: U is the set of all users, I the set of all items and T the set of all tags. The historical tagging information is given by $S \subseteq U \times I \times T$. As this is a ternary relation over categorical variables, it can be seen as a three-dimensional tensor (see figure 1) where the triples in S are the positive observations in the past. For tag recommendation, we are interested in recommending for a given user-item pair (u, i) a list of tags. Following [7], we call such a combination (u, i) a *post* and we define the set of all observed posts P_S:

$$P_S := \{(u, i) | \exists t \in T : (u, i, t) \in S\}$$

P_S can be seen as a two-dimensional projection of S on the user/item dimension using the OR operation.

Recommendation of tags for a given post (u, i) can be formulated as a ranking problem and thus as predicting a total order $>_{u,i} \subset T \times T$ over tags. That means each ranking $>_{u,i}$ has to satisfy:

$$\forall t_1, t_2 \in T : t_1 \neq t_2 \Rightarrow t_1 >_{u,i} t_2 \vee t_2 >_{u,i} t_1 \quad (1)$$

$$\forall t_1, t_2 \in T : t_1 >_{u,i} t_2 \wedge t_2 >_{u,i} t_1 \Rightarrow t_1 = t_2 \quad (2)$$

$$\forall t_1, t_2, t_3 \in T : t_1 >_{u,i} t_2 \wedge t_2 >_{u,i} t_3 \Rightarrow t_1 >_{u,i} t_3 \quad (3)$$

where (1) is totality, (2) is antisymmetry and (3) is transitivity. All of the models presented in this paper predict a scoring function $\hat{Y} : U \times I \times T \to \mathbb{R}$ which can be used to derive an order that trivially satisfies antisymmetry and transitivity. If the scoring function gives an identical score for two different tags and the same user-item combination, we place randomly one of the tags before the other – this ensures totality.

Often the number of predicted tags should be restricted. We therefore also define the list of the Top-N tags as:

$$\text{Top}(u, i, N) := \underset{t \in T}{\overset{N}{\text{argmax}}} \hat{y}_{u,i,t} \quad (4)$$

with N being the number of tags in the target list.

3.2 Data Analysis

The main problem in data mining/ machine learning from data of a tagging system is that there are only observations S of positive tagging events (see figure 1). That means the system observes what tags a user likes to give for an item but not which tags he does not like to give. For applying machine learning (e.g. optimizing a objective criterion) usually also examples of such negative events are necessary. A common approach [22, 6, 16] is to place all triples that are not in S – i.e. $(U \times I \times T) \setminus S$ – in the negative class. This approach has several drawbacks which is discussed in detail in [18] for the task of tag recommendation.

Instead, we propose to infer pairwise ranking constraints D_S from S like in [18, 17]. The idea is that within a post (u, i), one can assume that a tag t_A is preferred over another tag t_B iff (u, i, t_A) has been observed and (u, i, t_B) has not been observed. An example is given in figure 2. In total, the training data D_S for pairwise constraints is defined as:

$$D_S := \{(u, i, t_A, t_B) : (u, i, t_A) \in S \wedge (u, i, t_B) \notin S\}$$

The main advantage of our approach is, that the rankings $>_{\cdot,\cdot}$ that should be predicted in the future are treated as missing values (see the '?'s figure 2). Other approaches like [22] learn that all these tags are not liked – i.e. they should have the same preference score 0. A more detailed discussion can be found in [17] for the related task of item recommendation.

4. BAYESIAN PERSONALIZED RANKING (BPR) FOR TAG RECOMMENDATION

In the following, we derive the optimization criterion BPR-Opt and the learning algorithm LearnBPR for tag recommendation that will later on be used to optimize the factorization models. Please note that both the optimization criterion and the learning algorithm are generic and are not limited to factorization models. The analysis of this section is closely related to the original derivation of BPR-Opt and LearnBPR that we have introduced in [17] for the related problem setting of item recommendation.

4.1 BPR Optimization Criterion

The problem of finding the best ranking $>_{u,i} \subset T \times T$ for a given post (u, i) can be formalized as maximizing the following probability:

$$p(\Theta| >_{u,i}) \propto p(>_{u,i} |\Theta)\, p(\Theta)$$

where Θ are the model parameters. Assuming independence of posts, this leads to the maximum a posterior (MAP) estimator of the model parameters:

$$\underset{\Theta}{\operatorname{argmax}} \prod_{(u,i)\in U\times I} p(>_{u,i} |\Theta)\, p(\Theta) \tag{5}$$

Next, we will analyse $p(>_{u,i} |\Theta)$ in detail and show how it can be estimated from the observed data. First of all, we assume pairwise independence of $p(t_A >_{u,i} t_B|\Theta)$ and $p(t_C >_{u,i} t_D|\Theta)$ where $t_A \neq t_C$ and $t_B \neq t_D$. And as $t_A >_{u,i} t_B$ is a Bernoulli experiment, we can write:

$$\begin{aligned}&\prod_{(u,i)\in U\times I} p(>_{u,i} |\Theta)\\ &= \prod_{(u,i,t_A,t_B)\in U\times I\times T^2} p(t_A >_{u,i} t_B|\Theta)^{\delta((u,i,t_A,t_B)\in D_S)}\\ &\qquad \cdot (1 - p(t_A >_{u,i} t_B|\Theta))^{\delta((u,i,t_B,t_A)\in D_S)}\end{aligned}$$

with the indicator function δ:

$$\delta(b) := \begin{cases} 1 & \text{if } b \text{ is true,} \\ 0 & \text{else} \end{cases}$$

As the target function has to be a total order, this can be simplified to:

$$\prod_{(u,i)\in U\times I} p(>_{u,i} |\Theta) = \prod_{(u,i,t_A,t_B)\in D_S} p(t_A >_{u,i} t_B|\Theta) \tag{6}$$

Next, we derive an estimator for $p(t_A >_{u,i} t_B|\Theta)$ by plugging in a model $\hat{Y} : U \times I \times T^2 \to \mathbb{R}$ that relies on the model parameters Θ:

$$p(t_A >_{u,i} t_B|\Theta) := \sigma(\hat{y}_{u,i,t_A,t_B}(\Theta)) \tag{7}$$

where σ is the logistic function $\sigma(x) := \frac{1}{1+e^{-x}}$. To shorten notation, we will write $\hat{y}_{u,i,t_A,t_B}$ for $\hat{y}_{u,i,t_A,t_B}(\Theta)$.[1] In total, we have:

$$\prod_{(u,i)\in U\times I} p(>_{u,i} |\Theta) = \prod_{(u,i,t_A,t_B)\in D_S} \sigma(\hat{y}_{u,i,t_A,t_B}) \tag{8}$$

For the prior $p(\Theta)$, we assume that the model parameters are drawn from a Normal distribution $\Theta \sim N(0, \sigma_\Theta^2 I)$ centered at 0 and with σ_Θ being the model specific variance vector.

Filling this into the MAP estimator (5), we get the optimization criterion BPR-Opt for Bayesian Personalized Ranking:

$$\begin{aligned}\text{BPR-Opt} &:= \ln \prod_{(u,i,t_A,t_B)\in D_S} \sigma(\hat{y}_{u,i,t_A,t_B})\, p(\Theta)\\ &= \sum_{(u,i,t_A,t_B)\in D_S} \ln \sigma(\hat{y}_{u,i,t_A,t_B}) - \lambda_\Theta ||\Theta||_F^2\end{aligned}$$

where λ_Θ is the regularization constant corresponding to σ_Θ.

A more detailed discussion of BPR for the related problem of item recommendation can be found in [17]. There also the relationship to AUC optimization (like in [18]) is shown.

4.2 BPR Learning Algorithm

Secondly, we derive a learning algorithm to optimize the model parameters Θ of $\hat{y}_{u,i,t_A,t_B}$ for BPR-Opt. In general, optimizing BPR-Opt is very time consuming, as D_S is very large. The size of D_S is in $O(|S|\,|T|)$. E.g. for the examples of our evaluation section this would be about $3,299,006,344$ quadruples for the ECML/PKDD Discovery Challenge 09 and $449,290,590$ quadruples for our Last.fm subset. Thus computing the full gradients is very slow and normal gradient descent is not feasible. Also stochastic gradient descent where the quadruples are traversed in a sorted way like per post or per user will be slow – an example for this can be found in [17]. Instead, the BPR algorithm draws quadruples randomly from D_S. This is motivated by the observation that many quadruples overlap in three dimensions – i.e. for a post (u, i) with the positive tags t_1 and t_2, D_S includes the cases $(u, i, t_1, t_3), \ldots, (u, i, t_1, t_{|T|})$ and $(u, i, t_2, t_3), \ldots, (u, i, t_2, t_{|T|})$. This means that drawing a case randomly and performing stochastic gradient descent on the drawn case will also help many other related cases. In all, our generic learning algorithm LearnBPR for optimizing BPR-Opt for tag recommendation is shown in figure 3. The gradient of BPR-Opt given a case (u, i, t_A, t_B) with respect to a model parameter θ is:

$$\begin{aligned}&\frac{\partial}{\partial\theta}\left(\ln \sigma(\hat{y}_{u,i,t_A,t_B} - \lambda_\Theta ||\Theta||_F^2\right)\\ &\propto\ (1 - \sigma(\hat{y}_{u,i,t_A,t_B})) \cdot \frac{\partial}{\partial\theta}\hat{y}_{u,i,t_A,t_B} - \lambda_\theta\theta\end{aligned}$$

[1] Throughout this work, we use models where $\hat{y}_{u,i,t_A,t_B} := \hat{y}_{u,i,t_A} - \hat{y}_{u,i,t_B}$. But for BPR-Opt and LearnBPR this limitation is not necessary and thus we discuss the more general form of $\hat{y}_{u,i,t_A,t_B}$.

That means, to apply LearnBPR to a given model, only the gradient $\frac{\partial}{\partial\theta}\hat{y}_{u,i,t_A,t_B}$ has to be computed. In the next section, we derive our factorization models and also show their gradients for optimization w.r.t. BPR-Opt with LearnBPR.

1: **procedure** LearnBPR(D_S, Θ)
2: initialize Θ
3: **repeat**
4: draw (u, i, t_A, t_B) uniformly from D_S
5: $\Theta \leftarrow \Theta + \alpha \frac{\partial}{\partial\Theta}\left(\ln\sigma(\hat{y}_{u,i,t_A,t_B}) - \lambda_\Theta||\Theta||_F^2\right)$
6: **until** convergence
7: **return** $\hat{\Theta}$
8: **end procedure**

Figure 3: Optimizing tag recommender models for BPR with bootstrapping based stochastic gradient descent. With learning rate α and regularization λ_Θ.

5. FACTORIZATION MODELS

Factorization models are a very successful model class for recommender systems. E.g. many of the best performing models [10, 11] on the Netflix Challenge[2] for rating prediction are based on matrix factorization. Also for the related task of item prediction, factorization models are known [20, 6, 16, 17] to outperform models like k-nearest-neighbour collaborative filtering or the Bayesian models URP [15] and PLSA [4]. Also for tag recommendation recent results [18, 22] indicate that factorization models generate high quality predictions outperforming other approaches like Folkrank and adapted Pagerank [7]. In contrast to factorization models in two dimensions (matrix factorization), in tag recommendation there are many possibilities for factorizing the data. To the best of our knowledge, in tag recommendation only models based on Tucker decomposition have been analyzed yet [18, 22].

In the following, we describe three factorization models for tag recommendation: Tucker decomposition (TD), Canonical decomposition (DC) and our pairwise interaction tensor factorization model (PITF) (see figure 4). We will show for each model how it can be learned with BPR and the relationships to the other models.

All of our factorization models predict a scoring function $\hat{Y} : U \times I \times T \to \mathbb{R}$ which can be seen as a three-dimensional tensor Y where the value of entry (u, i, t) is the score $\hat{y}_{u,i,t}$. That means for ranking within a post, we sort the tags with respect to $\hat{y}_{u,i,t}$. And thus for applying BPR optimization, we set:

$$\hat{y}_{u,i,t_A,t_B} := \hat{y}_{u,i,t_A} - \hat{y}_{u,i,t_B}$$

5.1 Tucker Decomposition (TD) model

Tucker Decomposition [23] factorizes a higher-order cube into a core tensor and one factor matrix for each dimensions.

$$\hat{y}^{\mathrm{TD}}_{u,i,t} := \sum_{\tilde{u}}\sum_{\tilde{i}}\sum_{\tilde{t}} \hat{c}_{\tilde{u},\tilde{i},\tilde{t}} \cdot \hat{u}_{u,\tilde{u}} \cdot \hat{i}_{i,\tilde{i}} \cdot \hat{t}_{t,\tilde{t}} \tag{9}$$

or equivalently as tensor product (see figure 4):

$$\hat{Y}^{\mathrm{TD}} := \hat{C} \times_u \hat{U} \times_i \hat{I} \times_t \hat{T} \tag{10}$$

[2] http://www.netflixprize.com/

with model parameters:

$$\hat{C} \in \mathbb{R}^{k_u \times k_i \times k_t}, \quad \hat{U} \in \mathbb{R}^{|U| \times k_u}$$
$$\hat{I} \in \mathbb{R}^{|I| \times k_i}, \quad \hat{T} \in \mathbb{R}^{|T| \times k_t}$$

For learning such a TD model with BPR-Opt, the gradients $\frac{\partial \hat{Y}^{\mathrm{TD}}}{\partial\hat{\Theta}}$ are:

$$\frac{\partial \hat{y}^{\mathrm{TD}}_{u,i,t}}{\partial \hat{c}_{\tilde{u},\tilde{i},\tilde{t}}} = \hat{u}_{u,\tilde{u}} \cdot \hat{i}_{i,\tilde{i}} \cdot \hat{t}_{t,\tilde{t}}$$
$$\frac{\partial \hat{y}^{\mathrm{TD}}_{u,i,t}}{\partial \hat{u}_{u,\tilde{u}}} = \sum_{\tilde{i}}\sum_{\tilde{t}} \hat{c}_{\tilde{u},\tilde{i},\tilde{t}} \cdot \hat{i}_{i,\tilde{i}} \cdot \hat{t}_{t,\tilde{t}}$$
$$\frac{\partial \hat{y}^{\mathrm{TD}}_{u,i,t}}{\partial \hat{i}_{i,\tilde{i}}} = \sum_{\tilde{u}}\sum_{\tilde{t}} \hat{c}_{\tilde{u},\tilde{i},\tilde{t}} \cdot \hat{u}_{u,\tilde{u}} \cdot \hat{t}_{t,\tilde{t}}$$
$$\frac{\partial \hat{y}^{\mathrm{TD}}_{u,i,t}}{\partial \hat{t}_{t,\tilde{t}}} = \sum_{\tilde{u}}\sum_{\tilde{i}} \hat{c}_{\tilde{u},\tilde{i},\tilde{t}} \cdot \hat{u}_{u,\tilde{u}} \cdot \hat{i}_{i,\tilde{i}}$$

An obvious drawback of TD is that the model equation is a nested sum of degree 3 – i.e. it is cubic in $k := \min(k_u, k_i, k_t)$ and so the runtime complexity for predicting one triple (u, i, t) is $O(k^3)$. Thus learning a TD model is slow even for a small to mid-sized number of factorization dimensions.

5.2 Canonical Decomposition (CD) model

The CD model (Canonical Decomposition) is a special case of the general Tucker Decomposition model.

$$\hat{y}^{\mathrm{CD}}_{u,i,t} := \sum_{f}^{k} \hat{u}_{u,f} \cdot \hat{i}_{i,f} \cdot \hat{t}_{t,f} \tag{11}$$

It can be derived from the Tucker Decomposition model by setting $\hat{C}$ to the diagonal tensor:

$$\hat{c}_{\tilde{u},\tilde{i},\tilde{t}} = \begin{cases} 1, & \text{if } \tilde{u} = \tilde{i} = \tilde{t} \\ 0, & \text{else} \end{cases}$$

Obviously, only the first $k := \min\{k_u, k_i, k_t\}$ features are used — i.e. if the dimensionality of the feature matrices differ, some features are not used, as the core will be 0 for these entries.

The gradients for this model are:

$$\frac{\partial \hat{y}^{\mathrm{CD}}_{u,i,t}}{\partial \hat{u}_{u,f}} = \hat{i}_{i,f} \cdot \hat{t}_{t,f}$$
$$\frac{\partial \hat{y}^{\mathrm{CD}}_{u,i,t}}{\partial \hat{i}_{i,f}} = \hat{u}_{u,f} \cdot \hat{t}_{t,f}$$
$$\frac{\partial \hat{y}^{\mathrm{CD}}_{u,i,t}}{\partial \hat{t}_{t,f}} = \hat{u}_{u,f} \cdot \hat{i}_{i,f}$$

Obviously, the CD model has a much better runtime complexity as the model equation contains no nested sums and thus is in $O(k)$.

5.3 Pairwise Interaction Tensor Factorization (PITF) model

Our approach explicitly models the two-way interactions between users, tags and items by factorizing each of the three relationships:

$$\hat{y}_{u,i,t} = \sum_{f} \hat{u}^T_{u,f} \cdot \hat{t}^U_{t,f} + \sum_{f} \hat{i}^T_{i,f} \cdot \hat{t}^I_{t,f} + \sum_{f} \hat{u}^I_{u,f} \cdot \hat{i}^U_{i,f} \tag{12}$$

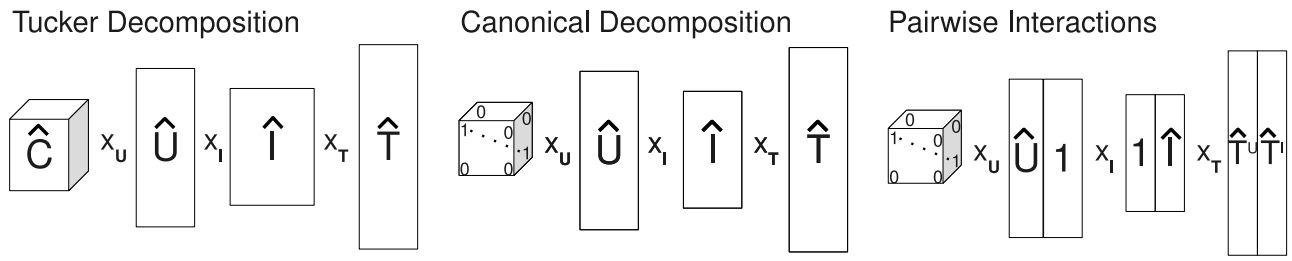


Figure 4: Tensor Factorization models: $\hat{C}$, $\hat{U}$, $\hat{I}$ and $\hat{T}$ are the model parameters (one tensor, three matrices). In Tucker Decomposition the core $\hat{C}$ is variable and the factorization dimensions can differ. For Canonical Decomposition and Pairwise Interactions the core is a fixed diagonal tensor. In Pairwise Interaction parts of the feature matrices are fixed which corresponds modelling pairwise interactions.

The user-item interaction vanishes for predicting rankings and for BPR optimization. The reason is that given a post (u, i), both the optimization criterion BPR and the ranking ignores any score on the user-item interaction. This results in our final model equation that we will refer to as the PITF (Pairwise Interaction Tensor Factorization) model:

$$\hat{y}_{u,i,t} = \sum_f \hat{u}_{u,f} \cdot \hat{t}^U_{t,f} + \sum_f \hat{i}_{i,f} \cdot \hat{t}^I_{t,f} \tag{13}$$

with model parameters:

$$\hat{U} \in \mathbb{R}^{|U| \times k}, \quad \hat{I} \in \mathbb{R}^{|I| \times k},$$
$$\hat{T}^U \in \mathbb{R}^{|T| \times k}, \quad \hat{T}^I \in \mathbb{R}^{|T| \times k}$$

Again, the runtime for predicting a triple (u, i, t) is in $O(k)$.

PITF is a special case of the CD model with dimensionality $2 \cdot k$ where:

$$\hat{u}^{\text{CD}}_{u,f} = \begin{cases} \hat{u}_{u,f}, & \text{if } f \leq k \\ 1, & \text{else} \end{cases}$$

$$\hat{i}^{\text{CD}}_{i,f} = \begin{cases} 1, & \text{if } f \leq k \\ \hat{i}_{i,f-k}, & \text{else} \end{cases}$$

$$\hat{t}^{\text{CD}}_{u,f} = \begin{cases} \hat{t}^U_{t,f}, & \text{if } f \leq k \\ \hat{t}^I_{t,f-k}, & \text{else} \end{cases}$$

The gradients for the PITF model are:

$$\frac{\partial \hat{y}_{u,i,t}}{\partial \hat{u}_{u,f}} = \hat{t}^U_{t,f}, \quad \frac{\partial \hat{y}_{u,i,t}}{\partial \hat{i}_{i,f}} = \hat{t}^I_{t,f},$$
$$\frac{\partial \hat{y}_{u,i,t}}{\partial \hat{t}^U_{t,f}} = \hat{u}_{u,f}, \quad \frac{\partial \hat{y}_{u,i,t}}{\partial \hat{t}^I_{t,f}} = \hat{i}_{u,f}$$

The complete BPR learning algorithm for PITF can be found in figure 5.

5.4 Relation between TD, CD and PITF

We have shown the relationships of our proposed PITF model to both the CD and TD model class. Obviously, the expressiveness of the model classes is:

$$\mathcal{M}^{\text{TD}} \supset \mathcal{M}^{\text{CD}} \supset \mathcal{M}^{\text{PITF}}$$

At first glance, one might think that reducing the expressiveness leads to worse prediction quality — i.e. that quality is traded in for e.g. runtime. But actually, our evaluation shows that this is not always the case. The reason is that our PI approach explicitly models a structure that might be hard to find for the TD and CD approach. Especially, regularization approaches like ridge regression which usually assume that the model parameters are normally distributed with mean zero $\Theta \sim N(0, \sigma^2_\Theta I)$ 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.

1: **procedure** LEARNBPR-PITF($P_S, \hat{U}, \hat{I}, \hat{T}^U, \hat{T}^I$)
2: draw $\hat{U}, \hat{I}, \hat{T}^U, \hat{T}^I$ from $N(\mu, \sigma^2)$
3: **repeat**
4: draw (u, i, t_A, t_B) uniformly from D_S
5: $\hat{y}_{u,i,t_A,t_B} \leftarrow \hat{y}_{u,i,t_A} - \hat{y}_{u,i,t_B}$
6: $\delta \leftarrow (1 - \sigma(\hat{y}_{u,i,t_A,t_B}))$
7: **for** $f \in 1, \ldots, k$ **do**
8: $\hat{u}_{u,f} \leftarrow \hat{u}_{u,f} + \alpha \left(\delta \cdot (\hat{t}^U_{t_A,f} - \hat{t}^U_{t_B,f}) - \lambda \cdot \hat{u}_{u,f}\right)$
9: $\hat{i}_{i,f} \leftarrow \hat{i}_{i,f} + \alpha \left(\delta \cdot (\hat{t}^I_{t_A,f} - \hat{t}^I_{t_B,f}) - \lambda \cdot \hat{i}_{i,f}\right)$
10: $\hat{t}^U_{t_A,f} \leftarrow \hat{t}^U_{t_A,f} + \alpha \left(\delta \cdot \hat{u}_{u,f} - \lambda \cdot \hat{t}^U_{t_A,f}\right)$
11: $\hat{t}^U_{t_B,f} \leftarrow \hat{t}^U_{t_B,f} + \alpha \left(-\delta \cdot \hat{u}_{u,f} - \lambda \cdot \hat{t}^U_{t_B,f}\right)$
12: $\hat{t}^I_{t_A,f} \leftarrow \hat{t}^I_{t_A,f} + \alpha \left(\delta \cdot \hat{i}_{i,f} - \lambda \cdot \hat{t}^I_{t_A,f}\right)$
13: $\hat{t}^I_{t_B,f} \leftarrow \hat{t}^I_{t_B,f} + \alpha \left(-\delta \cdot \hat{i}_{i,f} - \lambda \cdot \hat{t}^I_{t_B,f}\right)$
14: **end for**
15: **until** convergence
16: **return** $\hat{U}, \hat{I}, \hat{T^U}, \hat{T^I}$
17: **end procedure**

Figure 5: Optimizing the PITF model with LearnBPR.

6. EVALUATION

In our evaluation, we investigate the learning runtime and prediction quality of our proposed PITF model. For the runtime, we want to justify the results of the theoretical complexity analysis (TD is in O(k^3), CD/PITF in O(k)) by an empirical comparison of the TD model to the CD/PARAFAC model and our PITF model. With respect to prediction quality, we investigate empirically whether the speedup of CD/ PITF is paid with quality – i.e. if there is a trade-off between quality and runtime between the model classes.

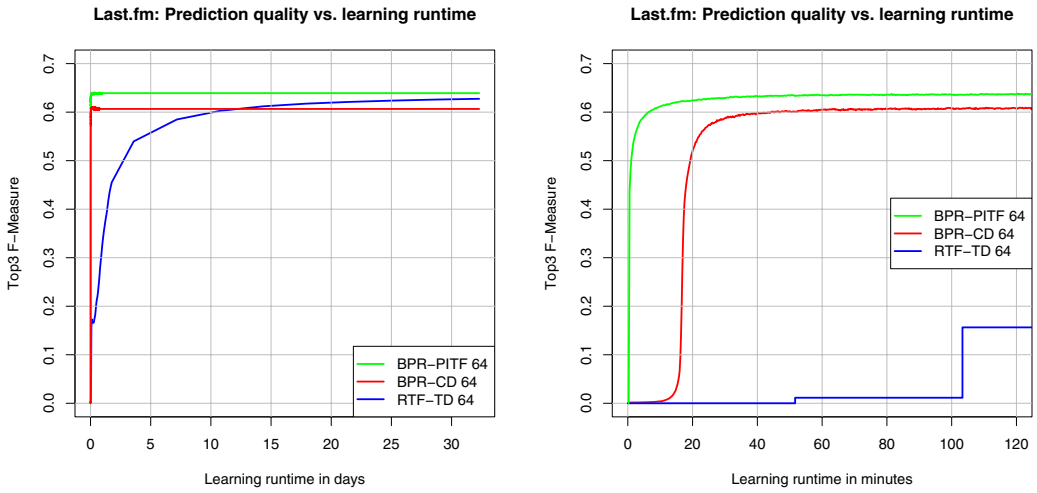


Figure 6: F-Measure on top-3 list after training a model for x days/ hours. Learning a high quality TD model (RTF-TD [18]) on a larger dataset like Last.fm takes several days. The PITF and CD models give good prediction quality already after 20 and 40 minutes respectively.

6.1 Datasets

We use three datasets for evaluation: Bibsonomy and Last.fm like in [7, 18] and the dataset from the ECML/ PKDD Discovery Challenge 2009[3]. All datasets are p-cores[4] – for BibSonomy the 5-core, for Last.fm the 10-core and for the ECML/PKDD Challenge the provided 2-core. The characteristics of the datasets can be found in table 1.

6.2 Evaluation Methodology

For Bibsonomy and Last.fm we use the same protocol as described in [8, 18] – i.e. per user one post is randomly removed from the training set S_{train} and put into the test set S_{test}. We use the exactly same splits as in [18]. For the ECML Challenge dataset we randomly remove overall 1,185 posts, and put them into the test set – the reason is that this dataset contains many users that only have 2 posts. Furthermore on ECML, we only removed such posts that the training data remains a 2-core.

After the splits have been built, the recommenders are trained on the test set and then the prediction quality on the test set is measured. We use the common evaluation scheme of F-measure in TopN-lists.

$$\text{Prec}(S_{test}, N) := \underset{(u,i)\in P_{S_{test}}}{\text{avg}} \frac{|\,\text{Top}(u,i,N) \cap \{t|(u,i,t) \in S_{test}\}|}{N}$$

$$\text{Rec}(S_{test}, N) := \underset{(u,i)\in P_{S_{test}}}{\text{avg}} \frac{|\,\text{Top}(u,i,N) \cap \{t|(u,i,t) \in S_{test}\}|}{|\{t|(u,i,t) \in S_{test}\}|}$$

$$\text{F1}(S_{test}, N) := \frac{2 \cdot \text{Prec}(S_{test}, N) \cdot \text{Rec}(S_{test}, N)}{\text{Prec}(S_{test}, N) + \text{Rec}(S_{test}, N)}$$

The experiments are repeated 10 times by sampling new training/ test sets. We report the average over all runs. The reported f-measure is the f-measure over the average recall and average precision.

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

[4] The p-core of S is the largest subset of S with the property that every user, every item and every tag has to occur in at least p posts.

The hyperparameters of all models are searched on the first training split. For the RTF-TD and HOSVD model the hyperparameters are the same as in [18]. For PITF the hyperparameters are $\lambda = 5e-05$ and $\alpha = 0.05$. For CD they are $\lambda = 0$ and $\alpha = 0.01$. The parameters of both models were initialized with $N(0, 0.01)$.

The runtime measurements of RTF-TD, BPR-PITF and BPR-CD were made with C++ implementations. The experiments were run on a compute cluster with 200 cores in total. Each compute node has identical hard- and software. Our C++ implementations use no parallelization neither over compute nodes nor within nodes – i.e. per run only one core was used.

Furthermore, we compare to other recent tag recommender methods: HOSVD [22], FolkRank and Adapted Pagerank [5] as well as the upper bound for non-personalized tag recommenders [18].

6.3 Results

Learning runtime.

The comparison of the convergence of BPR-PITF to BPR-CD and RTF-TD on the Last.fm dataset can be found in figure 6. Here you can see how the prediction quality improves after training a model (k=64) for a given time span. The left chart shows the quality over a span of 30 days. RTF-TD needs about 12 days to achieve a prediction quality as good as BPR-CD. Even after 30 days of training, the quality of RTF-TD is still worse than BPR-PITF.

In contrast to this, BPR-PITF and BPR-CD converge much faster. The right chart shows the quality over the first two hours. BPR-PITF and BPR-CD achieve convergence already after 20 and 40 minutes respectively. As each iteration of RTF-TD takes more than 50 minutes, the progress is very slow. When comparing BPR-PITF and BPR-CD among each other, one can see, that BPR-PITF converges faster. It is interesting to see that in the beginning BPR-CD seems to need several updates (18 minutes) before the

dataset	Users $\|U\|$	Items $\|I\|$	Tags $\|T\|$	Triples $\|S\|$	Posts $\|P_S\|$
BibSonomy	116	361	412	10,148	2,522
Last.fm	2,917	1,853	2,045	219,702	75,565
ECML/PKDD Discovery Challenge 09	1,185	22,389	13,276	248,494	63,628

Table 1: Dataset characteristics in terms of number of users, items, tags, tagging triples S and posts.

quality improves reasonably. One explanation could be that BPR-CD is searching the structure among the three-way interactions whereas in BPR-PITF this is already given by the two pairwise interactions.

The worse empirical runtime results of RTF-TD in comparison to BPR-CD and BPR-PITF match to the theoretical runtime complexity analysis of the model equations (see section 5). Furthermore, learning for both BPR-CD and BPR-PITF can be easily parallelized because quadruples of two draws usually share no parameters – in contrast to this, all entries in RTF-TD share the core tensor which makes it more difficult to parallelize RTF-TD.

Prediction quality.

Secondly, we compare the prediction quality of BPR-PITF to competing models. In figure 8, a comparison to BPR-CD, RTF-TD, Folkrank, Pagerank and HOSVD on Bibsonomy and Last.fm is shown. In general, the factorization models result in the best prediction quality – only on the very small Bibsonomy dataset Folkrank is competitive.

When comparing the two factorization models with linear runtime in k – i.e. CD and PITF – one can see that BPR-PITF achieves on all datasets a higher prediction quality than BPR-CD. At first, this might be surprising because CD is more general and includes PITF. It seems that BPR-CD is unable to find the pairwise structure of PITF and to do regularization at the same time. An indication for this is that for CD the 'best' regularization parameter found by grid search is $\lambda = 0$.

Next, we compare the prediction quality of the pairwise interaction model to full Tucker decomposition. On the small Bibsonomy dataset, on small TopN-lists (1,2,3) RTF-TD outperforms BPR-PITF whereas on larger lists, the difference vanishes. In contrast to this on the larger Last.fm dataset BPR-PITF outperforms RTF-TD on all list sizes. These results indicate that the learning speedup of BPR-PITF models to RTF-TD does not come to the prize of lower prediction quality. Rather, BPR-PITF can even outperform RTF-TD in quality on larger datasets.

Finally, figure 9 shows the prediction quality of BPR-PITF with an increasing number of factorization dimensions from 8 to 256. As you can see, on all three datasets the prediction quality does not benefit from more than 64 dimensions.

ECML / PKDD Discovery Challenge 09.

In addition to the lab experiments, our BPR-PITF model took also part in task 2 of the ECML/PKDD Discovery Challenge 09 and achieved the highest prediction quality. Figure 7 shows the final results[5] listing the first six approaches. This evaluation in a tag recommender challenge organized by a third party shows that BPR-PITF is able to create high quality predictions.

[5] `http://www.kde.cs.uni-kassel.de/ws/dc09/results`

Rank	Method	Top-5 F-Measure
1	BPR-PITF + adaptive list size	0.35594
-	*BPR-PITF (not submitted)*	*0.345*
2	Relational Classification [14]	0.33185
3	Content-based [13]	0.32461
4	Content-based [25]	0.32230
5	Content-based [9]	0.32134
6	Personomy translation [24]	0.32124
...	...	...

Figure 7: Official results (top-6) from the ECML/ PKDD Discovery Challenge 2009.

Our approach at the ECML/PKDD Challenge had two additions to the BPR-PITF presented in this paper: (1) In the challenge, the recommender could benefit from suggesting lists with less than 5 tags – thus we estimated how many tags to recommend. Even without this enhancement for the challenge, our approach would still have the best score with 0.345. (2) We ensembled many BPR-PITF models to reduce variance in the ranking estimates. On our holdout test this improved the result only a little bit [19].

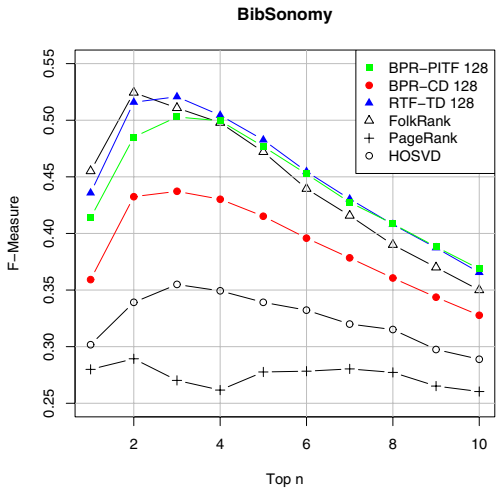
7. CONCLUSION AND FUTURE WORK

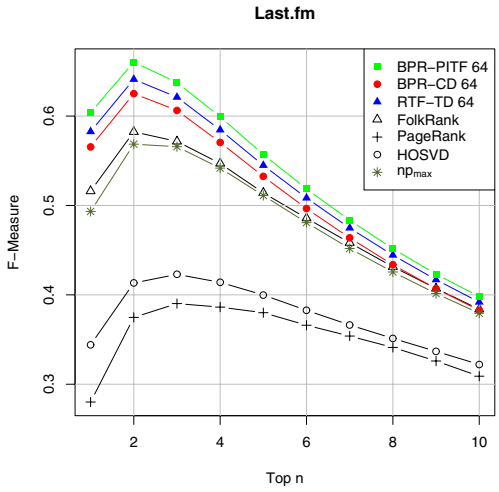
In this work we have presented a new factorization model for tag recommendation, that explicitly models the pairwise interactions (PITF) between users, items and tags. We have shown the relationships of our model to the Tucker decomposition (TD) and the Canonical descomposition (CD/PARAFAC). The advantage of our PITF model is the linear runtime in the factorization dimension whereas TD is cubic. Furthermore we have adapted the Bayesian Personalized Ranking (BPR) framework including the BPR optimization criterion and the BPR learning algorithm from the related field of item recommendation to tag recommendation. This BPR framework for tag recommendation is generic and not limited to optimizing our proposed models. Finally, we have empirically shown that our model PITF largely outperforms RTF-TD in learning runtime and achieves better prediction quality on datasets of large scale. The empirical comparison was done on lab experiments and on the 'ECML/ PKDD Discovery Challenge 2009', that PITF has won.

In future work, we want to investigate other regularization approaches for TD and CD/PARAFAC models that might be able to learn a 'better' model structure than our pairwise interaction model.

Acknowledgments

We would like to thank Christoph Freudenthaler for fruitful discussions and helpful comments on this work. Steffen Rendle is supported by a research fellowship of the Japan Society for the Promotion of Science (JSPS). This work is partially co-funded through the European Commission FP7 project MyMedia (www.mymediaproject.org) under the grant agreement no. 215006.





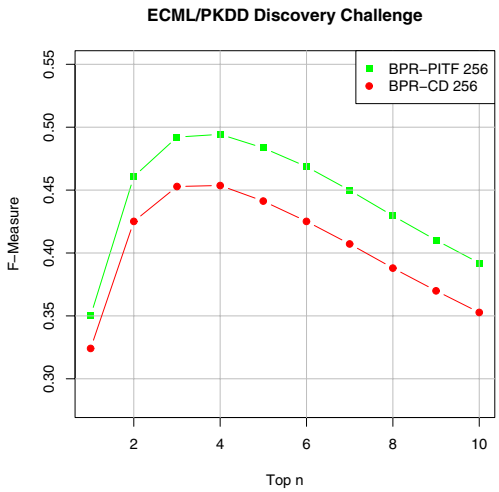
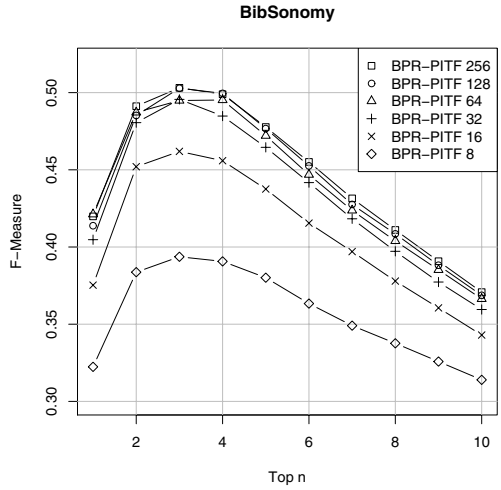
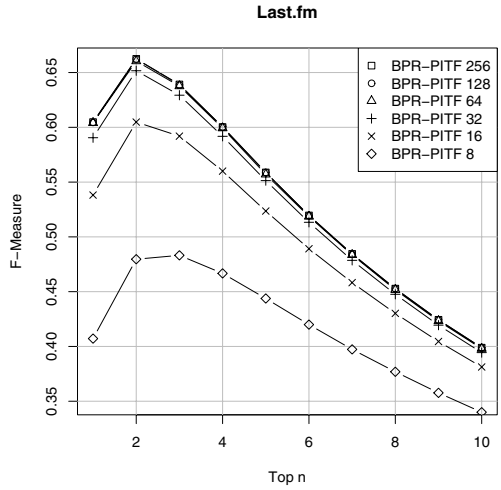


Figure 8: The tensor factorization models (RTF-TD, BPR-CD, BPR-PITF) achieve the best prediction quality outperforming other approaches like FolkRank, PageRank and HOSVD. On the larger datasets Last.fm and ECML/Discovery Challenge 09 the BPR-PITF model has the highest quality.





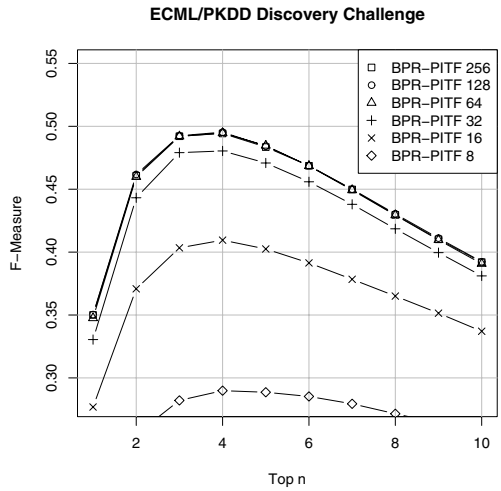


Figure 9: Quality comparison of BPR-PITF with an increasing number of factorization dimensions. On Last.fm and the Challenge dataset, there is no further improvement with more than 64 dimensions.

8. REFERENCES

[1] J. Carroll and J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. *Psychometrika*, 35:283–319, 1970.

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

[3] P. Heymann, D. Ramage, and H. Garcia-Molina. Social tag prediction. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 531–538. ACM, 2008.

[4] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

[5] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In Y. Sure and J. Domingue, editors, *The Semantic Web: Research and Applications*, volume 4011 of *Lecture Notes in Computer Science*, pages 411–426, Heidelberg, June 2006. Springer.

[6] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *IEEE International Conference on Data Mining (ICDM 2008)*, pages 263–272, 2008.

[7] R. Jaeschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), Warsaw, Poland*, 2007.

[8] R. Jaeschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in social bookmarking systems. *AICOM*, 2008.

[9] S. Ju and K.-B. Hwang. A weighting scheme for tag recommendation in social bookmarking systems. In *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2009.

[10] 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*, pages 426–434, New York, NY, USA, 2008. ACM.

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

[12] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000.

[13] M. Lipczak, Y. Hu, Y. Kollet, and E. Milios. Tag sources for recommendation in collaborative tagging systems. In *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2009.

[14] L. B. Marinho, C. Preisach, and L. Schmidt-Thieme. Relational classification for personalized tag recommendation. In *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2009.

[15] B. Marlin. Modeling user rating profiles for collaborative filtering. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.

[16] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. M. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *IEEE International Conference on Data Mining (ICDM 2008)*, pages 502–511, 2008.

[17] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI 2009)*, 2009.

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

[19] S. Rendle and L. Schmidt-Thieme. Factor models for tag recommendation in bibsonomy. In *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2009.

[20] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 713–719, New York, NY, USA, 2005. ACM.

[21] Y. Song, L. Zhang, and C. L. Giles. A sparse gaussian processes classification framework for fast tag suggestions. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 93–102. ACM, 2008.

[22] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos. Tag recommendations based on tensor dimensionality reduction. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 43–50, New York, NY, USA, 2008. ACM.

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

[24] R. Wetzker, A. Said1, and C. Zimmermann. Understanding the user: Personomy translation for tag recommendation. In *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2009.

[25] N. Zhang, Y. Zhang, and J. Tang. A tag recommendation system based on contents. In *Proceedings of the ECML-PKDD Discovery Challenge Workshop*, 2009.