Collaborative Filtering for Binary, Positive-only Data

Koen Verstrepen[*]
Froomle
Antwerp, Belgium
koen.verstrepen@froomle.com

Kanishka Bhaduri[†]
Apple, Inc.
Cupertino, USA
kanishka.bh@gmail.com

Boris Cule
University of Antwerp
Antwerp, Belgium
boris.cule@uantwerp.be

Bart Goethals
Froomle, University of Antwerp
Antwerp, Belgium
bart.goethals@uantwerp.be

ABSTRACT

Traditional collaborative filtering assumes the availability of explicit ratings of users for items. However, in many cases these ratings are not available and only binary, positive-only data is available. Binary, positive-only data is typically associated with implicit feedback such as items bought, videos watched, ads clicked on, etc. However, it can also be the results of explicit feedback such as likes on social networking sites. Because binary, positive-only data contains no negative information, it needs to be treated differently than rating data. As a result of the growing relevance of this problem setting, the number of publications in this field increases rapidly. In this survey, we provide an overview of the existing work from an innovative perspective that allows us to emphasize surprising commonalities and key differences.

1. INTRODUCTION

Increasingly, people are overwhelmed by an abundance of choice. Via the World Wide Web, everybody has access to a wide variety of news, opinions, (encyclopedic) information, social contacts, books, music, videos, pictures, products, jobs, houses, and many other *items*, from all over the world. However, from the perspective of a particular person, the vast majority of items is irrelevant; and the few relevant items are difficult to find because they are buried under a large pile or irrelevant ones. There exist, for example, lots of books that one would enjoy reading, if only one could identify them. Moreover, not only do people fail to find relevant existing items, niche items fail to be created because it is anticipated that the target audience will not be able to find them under the pile of irrelevant items. Certain books, for example, are never written because writers anticipate they will not be able to reach a sufficiently large portion of their target audience, although the audience exists. *Recommender systems* contribute to overcome these difficulties by *connecting* individuals with items relevant to them. A good book recommender system, for example, would typically recommend 3, previously unknown, books that the user would enjoy reading, and that are sufficiently different from each other. Studying recommender systems specifically, and the connection between individuals and relevant items in general, is the subject of *recommendation* research. But the relevance of recommendation research goes beyond connecting users with items. Recommender systems can, for example, also connect genes with diseases, biological targets with drug compounds, words with documents, tags with photos, etc.

[*]This work was done while Koen Verstrepen was working at the University of Antwerp.

[†]This work was done while Kanishka Bhaduri was working at Netflix, Inc.

1.1 Collaborative Filtering

Collaborative filtering is a principal problem in recommendation research. In the most abstract sense, collaborative filtering is the problem of weighting missing edges in a bipartite graph.

The concrete version of this problem that got most attention until recently is *rating prediction*. In rating prediction, one set of nodes in the bipartite graph represent *users*, the other set of nodes represent *items*, an edge with weight r between user u and item i expresses that u has given i the rating r, and the task is to predict the missing ratings. Since rating prediction is a mature domain, multiple overviews exist [23; 47; 1; 59]. Recently, the attention for rating prediction diminished because of multiple reasons. First, collecting rating data is relatively expensive in the sense that it requires a non-negligible effort from the users. Second, user ratings do not correlate as well with user behavior as one would expect. Users tend to give high ratings to items they think they should consume, for example a famous book by Dostoyevsky. However, they would rather read Superman comic books, which they rate much lower. Finally, in many applications, predicting ratings is not the final goal, and the predicted ratings are only used to find the most relevant items for every user. Consequently, high ratings need to be accurate whereas the exact value of low ratings is irrelevant. However, in rating prediction high and low ratings are equally important.

Today, attention is increasingly shifting towards collaborative filtering with *binary, positive-only data*. In this version, edges are unweighted, an edge between user u and item i expresses that user u has given positive feedback about item i, and the task is to attach to every missing edge between a user u and an item i a score that indicates the suitability of recommending i to u. Binary, positive-only data is typically associated with implicit feedback such as items bought, videos watched, songs listened to, books lent from a library, ads clicked on, etc. However, it can also be the result of explicit feedback, such as *likes* on social networking sites. As a result of the growing relevance of this problem setting, the number of publications in this field increases rapidly. In this survey, we provide an overview of the existing work on collaborative filtering with binary, positive-only data from an innovative perspective that allows us to emphasize surprising commonalities and key differences. To enhance the readability, we sometimes omit the specification 'binary, positive-only' and use the abbreviated term 'collaborative filtering'.

Besides the bipartite graph, five types of extra information can be available. First, there can be item content or item metadata. In the case of books, for example, the content is the full text of the book and the metadata can include the writer, the publisher, the year it was published etc. Methods that exclusively use this kind of information are typically classified as *content based*. Methods that combine this kind of information with a collaborative filtering method are typically classified as *hybrid*. Second, there can be user metadata such as gender, age, location, etc. Third, users can be connected with each other in an extra, unipartite graph. A typical example is a social network between the users. An analogous graph can exist for the items. Finally, there can be contextual information such as location, date, time, intent, company, device, etc. Exploiting information besides the bipartite graph, is out of the scope of this survey. Comprehensive discussions on exploiting information outside the user-item matrix have been presented [53; 72].

1.2 Relation to Other Domains

To emphasize the unique aspects of collaborative filtering, we highlight the commonalities and differences with two related data science problems: classification and association rule mining.

First, collaborative filtering is equivalent to jointly solving many one-class classification problems, in which every one-class classification problem corresponds to one of the items. In the classification problem that corresponds to item i, i serves as the class, all other items serve as the features, the users that have i as a known preference serve as labeled examples and the other users serve as unlabeled examples. Amazon.com, for example, has more than 200 million items in its catalog, hence solving the collaborative filtering problem for Amazon.com is equivalent to jointly solving more than 200 million one-class classification problems, which obviously requires a distinct approach. However, collaborative filtering is more than efficiently solving many one-class classification problems. Because they are tightly linked, *jointly* solving all classification problems allows for sharing information between them. The individual classification problems share most of their features; and while i serves as the class in one of the classification problems, it serves as a feature in all other classification problems.

Second, association rule mining also assumes bipartite, unweighted data and can therefore be applied to datasets used for collaborative filtering. Furthermore, recommending item i to user u can be considered as the application of the association rule $I(u) \rightarrow i$, with $I(u)$ the itemset containing the known preferences of u. However, the goals of association rule mining and collaborative filtering are different. If a rule $I(u) \rightarrow i$ is crucial for recommending i to u, but irrelevant on the rest of the data, giving the rule a high score is desirable for collaborative filtering, but typically not for association rule mining.

1.3 Outline

After the Preliminaries (Sec. 2), we introduce our framework (Sec. 3) and review the state of the art along the three dimensions of our framework: Factorization Models (Sec. 4), Deviation Functions (Sec. 5), and Minimization Algorithms (Sec. 6). Finally, we discuss the usability of methods for rating prediction (Sec. 7) and conclude (Sec. 8).

2. PRELIMINARIES

We introduced collaborative filtering as the problem of weighting missing edges in a bipartite graph. Typically, however, this bipartite graph is represented by its adjacency matrix, which is called the preference matrix.

Let $\mathcal{U}$ be a set of users and $\mathcal{I}$ a set of items. We are given a preference matrix with training data $\mathbf{R} \in \{0,1\}^{|\mathcal{U}|\times|\mathcal{I}|}$. $\mathbf{R}_{ui} = 1$ indicates that there is a known preference of user $u \in \mathcal{U}$ for item $i \in \mathcal{I}$. $\mathbf{R}_{ui} = 0$ indicates that there is no such information. Notice that the absence of information means that either there exists no preference or there exists a preference but it is not known.

Collaborative filtering methods compute for every user-item pair (u,i) a recommendation score $s(u,i)$ that indicates the suitability of recommending i to u. Typically, the user-item-pairs are (partially) sorted by their recommendation scores. We define the matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ as $\mathbf{S}_{ui} = s(u,i)$. Furthermore, $c(x)$ gives the count of x, meaning

$$c(x) = \begin{cases} \sum_{i\in\mathcal{I}} R_{xi} & \text{if } x \in \mathcal{U} \\ \sum_{u\in\mathcal{U}} R_{ux} & \text{if } x \in \mathcal{I}. \end{cases}$$

Although we conveniently call the elements of $\mathcal{U}$ *users* and the elements of $\mathcal{I}$ *items*, these sets can contain any type of object. In the case of online social networks, for example, both sets contain the people that participate in the social network, i.e., $\mathcal{U} = \mathcal{I}$, and $\mathbf{R}_{ui} = 1$ if there exists a friendship link between persons u and i. In image tagging/annotation problems, $\mathcal{U}$ contains images, $\mathcal{I}$ contains words, and $\mathbf{R}_{ui} = 1$ if image u was tagged with word i. In chemogenomics, an early stage in the drug discovery process, $\mathcal{U}$ contains active drug compounds, $\mathcal{I}$ contains biological targets, and $\mathbf{R}_{ui} = 1$ if there is a strong interaction between compound u and biological target i.

Typically, datasets for collaborative filtering are extremely sparse, which makes it a challenging problem. The sparsity $\mathcal{S}$, computed as

$$\mathcal{S} = 1 - \frac{\sum_{(u,i)\in\mathcal{U}\times\mathcal{I}} \mathbf{R}_{ui}}{|\mathcal{U}| \cdot |\mathcal{I}|}, \tag{1}$$

typically ranges from 0.98 to 0.999 and is visualized in Figure 1. This means that a score must be computed for approximately 99% of the (u,i)-pairs based on only 1% of the (u,i)-pairs. This is undeniably a challenging task.

3. FRAMEWORK

In the most general sense, every method for collaborative filtering is defined as a function $\mathcal{F}$ that computes the recommendation scores $\mathbf{S}$ based on the data $\mathbf{R}$: $\mathbf{S} = \mathcal{F}(\mathbf{R})$. Since different methods $\mathcal{F}$ originate from different intuitions about the problem, they are typically explained from very

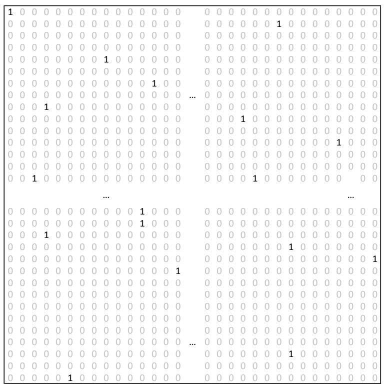

Figure 1: Typical sparsity of training data matrix $\mathbf{R}$ for binary, positive-only collaborative filtering.

different perspectives. In the literature on recommender systems in general and collaborative filtering specifically, two dominant perspectives have emerged: the model based perspective and the memory-based perspective. Unfortunately, these two are often described as two fundamentally separate classes of methods, instead of merely two perspectives on the same class of methods [47; 23]. As a result, the comparison between methods often remains superficial.

We, however, will explain many different collaborative filtering methods $\mathcal{F}$ from one and the same perspective. As such we facilitate the comparison and classification of these methods. Our perspective is a matrix factorization framework in which every method $\mathcal{F}$ consists of three fundamental building blocks: a factorization model of the recommendation scores $\mathbf{S}$, a deviation function that measures the deviation between the data $\mathbf{R}$ and the recommendation scores $\mathbf{S}$, and a minimization procedure that tries to find the model parameters that minimize the deviation function.

First, the **factorization model** computes the matrix of recommendation scores $\mathbf{S}$ as a link function l of a sum of T terms in which a term t is the product of F_t factor matrices:

$$\mathbf{S} = l\left(\sum_{t=1}^{T}\prod_{f=1}^{F_t}\mathbf{S}^{(t,f)}\right). \tag{2}$$

For many methods, l is the identity function, $T = 1$ and $F_1 = 2$. In this case, the factorization is given by:

$$\mathbf{S} = \mathbf{S}^{(1,1)}\mathbf{S}^{(1,2)}. \tag{3}$$

Because of their dimensions, $\mathbf{S}^{(1,1)} \in \mathbb{R}^{|\mathcal{U}|\times D}$ and $\mathbf{S}^{(1,2)} \in \mathbb{R}^{D\times|\mathcal{I}|}$ are often called the user-factor matrix and item-factor matrix, respectively. Figure 2 visualizes Equation 3. More complex models are often more realistic, but generally contain more parameters which increases the risk of overfitting.

Second, the number of terms T, the number of factor matrices F_t and the dimensions of the factor matrices are an integral part of the model, independent of the data $\mathbf{R}$[1]. Every entry in the factor matrices however, is a parameter that needs to be computed based on the data $\mathbf{R}$. We collectively denote these parameters as θ. Whenever we want to emphasize that the matrix of recommendation scores $\mathbf{S}$ is dependent on these parameters, we will write it as $\mathbf{S}(\theta)$. The model in Figure 2, for example, contains $(|\mathcal{U}| + |\mathcal{I}|) \cdot D$ parameters. Computing all the parameters in a factorization model is done by minimizing the *deviation* between the data $\mathbf{R}$ and the parameters θ. This deviation is measured by the **deviation function** $\mathcal{D}(\theta, \mathbf{R})$. Formally we compute the *optimal* values θ^* of the parameters θ as

$$\theta^* = \arg\min_{\theta} \mathcal{D}(\theta, \mathbf{R}).$$

Many deviation functions exist, and every deviation function mathematically expresses a different interpretation of the concept *deviation*.

Third, efficiently computing the parameters that minimize a deviation function is often non trivial because the majority of deviation functions is non-convex in the parameters of the factorization model. In that case, minimization algorithms can only compute parameters that correspond to a local minimum. The initialization of the parameters in the factorization model and the chosen hyperparameters of the minimization algorithm determine which local minimum will be found. If the value of the deviation function in this local minimum is not much higher than that of the global minimum, it is considered a good minimum. An intuitively appealing deviation function is worthless if there exists no algorithm that can efficiently compute parameters that correspond to a good local minimum of this deviation function.

Finally, we assume a basic usage scenario in which model parameters are recomputed periodically (typically, a few times a day). Computing the recommendation scores for a user based on the model, and extracting the items corresponding to the top-N scores, is assumed to be performed in *real time*. Specific aspects of scenarios in which models need to be recomputed in real time, are out of the scope of this survey.

In this overview, we first survey existing models in Section 4. Next, we compare the existing deviation functions in Section 5. Afterwards, we discuss the minimization algorithms that are used for fitting the model parameters to the deviation functions in Section 6. Finally, we discuss the applicability of rating-based methods in Section 7 and conclude in Section 8.

[1]High level statistics of the data might be taken into consideration to choose the model. There is however no clear, direct dependence on the data.

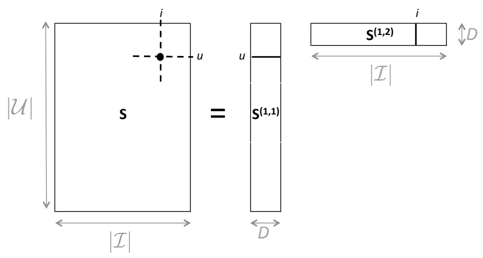

Figure 2: Matrix Factorization with 2 factor matrices (Eq. 3).

4. FACTORIZATION MODELS

Equation 2 gives a general description of all models for collaborative filtering. In this section, we discuss how the specific collaborative filtering models map to this equation.

4.1 Basic Models

A statistically well founded method is probabilistic latent semantic analysis (pLSA) by Hofmann, which is centered around the so called aspect model [19]. Hofmann models the probability $p(i|u)$ that a user u will prefer an item i as the mixture of D probability distributions induced by the hidden variables d:

$$p(i|u) = \sum_{d=1}^{D} p(i, d|u).$$

Furthermore, by assuming u and i conditionally independent given d, he obtains:

$$p(i|u) = \sum_{d=1}^{D} p(i|d) \cdot p(d|u).$$

This model corresponds to a basic two-factor matrix factorization:

$$\begin{aligned} \mathbf{S} &= \mathbf{S}^{(1,1)}\mathbf{S}^{(1,2)} \\ \mathbf{S}_{ui} &= p(i|u) \\ \mathbf{S}^{(1,1)}_{ud} &= p(d|u) \\ \mathbf{S}^{(1,2)}_{di} &= p(i|d), \end{aligned} \tag{4}$$

in which the $|\mathcal{U}| \times D$ parameters in $\mathbf{S}^{(1,1)}_{ud}$ and the $D \times |\mathcal{I}|$ parameters in $\mathbf{S}^{(1,2)}_{di}$ are a priori unknown and need to be computed based on the data $\mathbf{R}$. An appealing property of this model is the probabilistic interpretation of both the parameters and the recommendation scores. Fully in line with the probabilistic foundation, the parameters are constrained as:

$$\begin{aligned} \mathbf{S}^{(1,1)}_{ud} &\geq 0 \\ \mathbf{S}^{(1,2)}_{di} &\geq 0 \\ \sum_{i \in \mathcal{I}} p(i|d) &= 1 \\ \sum_{d=1}^{D} p(d|u) &= 1, \end{aligned} \tag{5}$$

expressing that both factor matrices are non-negative and that all row sums of $\mathbf{S}^{(1,1)}$ and $\mathbf{S}^{(1,2)}$ must be equal to 1 since they represent probabilities.

Latent dirichlet allocation (LDA) is a more rigorous statistical model, which puts Dirichlet priors on the parameters $p(d|u)$ [6; 19]. However, for collaborative filtering these priors are integrated out and the resulting model for computing recommendation scores is again a simple two factor factorization model.

The aspect model also has a geometric interpretation. In the training data $\mathbf{R}$, every user is profiled as a binary vector in an $|\mathcal{I}|$-dimensional space in which every dimension corresponds to an item. Analogously, every item is profiled as a binary vector in a $|\mathcal{U}|$-dimensional space in which every dimension corresponds to a user. Now, in the factorized representation, every hidden variable d represents a dimension of a D-dimensional space. Therefore, the matrix factorization $\mathbf{S} = \mathbf{S}^{(1,1)}\mathbf{S}^{(1,2)}$ implies a transformation of both user- and item-vectors to the same D-dimensional space. A row vector $\mathbf{S}^{(1,1)}_{u\cdot} \in \mathbb{R}^{1 \times D}$ is the representation of user u in this D-dimensional space and a column vector $\mathbf{S}^{(1,2)}_{\cdot i} \in \mathbb{R}^{D \times 1}$ is the representation of item i in this D-dimensional space. Figure 2 visualizes the user-vector of user u and the item-vector of item i. Now, as a result of the factorization model, a recommendation score $\mathbf{S}_{ui}$ is computed as the dotproduct of the user-vector of u with the item-vector of i:

$$\begin{aligned} \mathbf{S}^{(1,1)}_{u\cdot} \cdot \mathbf{S}^{(1,2)}_{\cdot i} &= \sum_{d=1}^{D} \mathbf{S}^{(1,1)}_{ud}\mathbf{S}^{(1,2)}_{di} \\ &= ||\mathbf{S}^{(1,1)}_{u\cdot}|| \cdot ||\mathbf{S}^{(1,2)}_{\cdot i}|| \cdot \cos(\phi_{ui}) \\ &= ||\mathbf{S}^{(1,2)}_{\cdot i}|| \cdot \cos(\phi_{ui}), \end{aligned} \tag{6}$$

with ϕ_{ui} the angle between the two vectors, and $||\mathbf{S}^{(1,1)}_{u\cdot}|| = 1$ a probabilistic constraint of the model (Eq.5). Therefore, an item i will be recommended if its vector norm $||\mathbf{S}^{(1,2)}_{\cdot i}||$ is large and ϕ_{ui}, the angle of its vector with the user vector, is small. From this geometric interpretation we learn that the recommendation scores computed with the model in Equation 4 contain both a personalized factor $\cos(\phi_{ui})$ and a non-personalized, popularity based factor $||\mathbf{S}^{(1,2)}_{\cdot i}||$.

Many other authors adopted this two-factor model. However, they abandoned its probabilistic foundation by removing the constraints on the parameters (Eq.5) [21; 36; 37; 55; 73; 45; 52; 61; 16; 12]:

$$\mathbf{S} = \mathbf{S}^{(1,1)}\mathbf{S}^{(1,2)}. \tag{7}$$

Yet another interpretation of this two-factor model is that every hidden variable d represents a cluster containing both users and items. A large value $\mathbf{S}^{(1,1)}_{ud}$ means that user u has a large degree of membership in cluster d. Similarly, a large value $\mathbf{S}^{(1,2)}_{di}$ means that item i has a large degree of membership in cluster d. As such, pLSA can be interpreted as a soft clustering model. According to the interpretation, item i will be recommended to a user u if they have high degrees of membership in the same clusters.

Although much less common, hard clustering models for collaborative filtering also exist. Hofmann and Puzicha [20] proposed the model

$$p(i|u, e(u) = c, d(i) = d) = p(i)\phi(e, d), \tag{8}$$

in which $e(u)$ indicates the cluster user u belongs to and $d(i)$ indicates the cluster item i belongs to. Furthermore, $\phi(e, d)$ is the association value between the user-cluster e and the item-cluster d. This cluster association factor increases or decreases the probability that u likes i relative to the independence model $p(i|u) = p(i)$. As opposed to the aspect model, this model assumes E user-clusters only containing users and D item-clusters only containing items. Furthermore a user or item belongs to exactly one cluster.

The factorization model corresponding to this approach is:

$$\begin{aligned} \mathbf{S} &= \mathbf{S}^{(1,1)}\mathbf{S}^{(1,2)}\mathbf{S}^{(1,3)}\mathbf{S}^{(1,4)}, \\ \mathbf{S}_{ui} &= p(i|u), \\ \mathbf{S}^{(1,1)}_{ue} &= \mathbb{I}(e(u) = e), \\ \mathbf{S}^{(1,2)}_{ed} &= \phi(e, d), \\ \mathbf{S}^{(1,3)}_{di} &= \mathbb{I}(d(i) = d), \\ \mathbf{S}^{(1,4)}_{ij} &= p(i) \cdot \mathbb{I}(i = j), \end{aligned}$$

in which $\mathbb{I}(true) = 1$, $\mathbb{I}(false) = 0$, and E and D are hyperparameters. The $|\mathcal{U}| \times E$ parameters in $\mathbf{S}^{(1,1)}$, $E \times D$ parameters in $\mathbf{S}^{(1,2)}$, $D \times |\mathcal{I}|$ parameters in $\mathbf{S}^{(1,3)}$ and the $|\mathcal{I}|$

parameters $\mathbf{S}^{(1,4)}$ need to be computed based on the data. Ungar and Foster [63] proposed a similar hard clustering method.

4.2 Explicitly Biased Models

In the above models, the recommendation score $\mathbf{S}_{ui}$ of an item i for a user u is the product of a personalized factor with an item-bias factor related to item-popularity. In Equation 6 the personalized factor is $\cos(\phi_{ui})$, and the bias factor is $||\mathbf{S}^{(1,2)}_{\cdot i}||$. In Equation 8 the personalized factor is $\phi(e,d)$, and the bias factor is $p(i)$. Other authors [28; 40; 24] proposed to model item- and user-biases as explicit terms instead of implicit factors. This results in the following factorization model:

$$\begin{aligned} \mathbf{S} &= \sigma\left(\mathbf{S}^{(1,1)} + \mathbf{S}^{(2,1)} + \mathbf{S}^{(3,1)}\mathbf{S}^{(3,2)}\right) \\ \mathbf{S}^{(1,1)}_{ui} &= b_u \qquad (9) \\ \mathbf{S}^{(2,1)}_{ui} &= b_i, \end{aligned}$$

with σ the sigmoid link-function, and $\mathbf{S}^{(1,1)}, \mathbf{S}^{(2,1)} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ the user- and item-bias matrices in which all columns of $\mathbf{S}^{(1,1)}$ are identical and also all rows of $\mathbf{S}^{(2,1)}$ are identical. The $|\mathcal{U}|$ parameters in $\mathbf{S}^{(1,1)}$, $|\mathcal{I}|$ parameters in $\mathbf{S}^{(2,1)}$, $|\mathcal{U}|\cdot D$ parameters in $\mathbf{S}^{(3,1)}$, and $|\mathcal{I}|\cdot D$ parameters in $\mathbf{S}^{(3,2)}$ need to be computed based on the data. D is a hyperparameter of the model. The goal of explicitly modeling the bias terms is to make the interaction term $\mathbf{S}^{(3,1)}\mathbf{S}^{(3,2)}$ a pure personalization term. Although bias terms are commonly used for collaborative filtering with rating data, only a few works with collaborative filtering with binary, positive-only data use them.

4.3 Basic Neighborhood Models

Multiple authors proposed special cases of the basic two-factor factorization in Equation 7.

4.3.1 Item-based

In a first special case, $\mathbf{S}^{(1,1)} = \mathbf{R}$ [10; 54; 45; 2; 34]. In this case, the factorization model is given by

$$\mathbf{S} = \mathbf{R}\mathbf{S}^{(1,2)}. \qquad (10)$$

Consequently, a user u is profiled by an $|\mathcal{I}|$-dimensional binary vector $\mathbf{R}_{u\cdot}$ and an item is profiled by an $|\mathcal{U}|$-dimensional real valued vector $\mathbf{S}^{(1,2)}_{\cdot i}$. This model is often interpreted as an *item-based neighborhood model* because the recommendation score $\mathbf{S}_{ui}$ of item i for user u is computed as

$$\mathbf{S}_{ui} = \mathbf{R}_{u\cdot}\cdot\mathbf{S}^{(1,2)}_{\cdot i} = \sum_{j\in\mathcal{I}} \mathbf{R}_{uj}\cdot\mathbf{S}^{(1,2)}_{ji} = \sum_{j\in I(u)} \mathbf{S}^{(1,2)}_{ji},$$

with $I(u)$ the known preferences of user u, and a parameter $\mathbf{S}^{(1,2)}_{ji}$ typically interpreted as the similarity between items j and i, i.e., $\mathbf{S}^{(1,2)}_{ji} = sim(j,i)$. Consequently, $\mathbf{S}^{(1,2)}$ is often called the item-similarity matrix. The SLIM method [34] adopts this model and additionally imposes the constraints

$$\mathbf{S}^{(1,2)}_{ji} \geq 0, \mathbf{S}^{(1,2)}_{ii} = 0. \qquad (11)$$

The non-negativity is imposed to enhance interpretability. The zero-diagonal is imposed to avoid finding a trivial solution for the parameters in which every item is maximally similar to itself and has zero similarity with any other item.

This model is rooted in the intuition that good recommendations are similar to the known preferences of the user. Although they do not present it as such, Gori et al. [18] proposed ItemRank, a method that is based on an interesting extension of this intuition: good recommendations are similar to other good recommendations, with a bias towards the known preferences of the user. The factorization model corresponding to ItemRank is based on PageRank [35] and given by:

$$\mathbf{S} = \alpha\cdot\mathbf{S}\mathbf{S}^{(1,2)} + (1-\alpha)\cdot\mathbf{R}. \qquad (12)$$

Because of the recursive nature of this model, $\mathbf{S}$ needs to be computed iteratively [18; 35].

4.3.2 User-based

Other authors proposed the symmetric counterpart of Equation 10 in which $\mathbf{S}^{(1,2)} = \mathbf{R}$ [48; 2; 3]. In this case, the factorization model is given by

$$\mathbf{S} = \mathbf{S}^{(1,1)}\mathbf{R}, \qquad (13)$$

which is often interpreted as a *user-based neighborhood model* because the recommendation score $\mathbf{S}_{ui}$ of item i for user u is computed as

$$\mathbf{S}_{ui} = \mathbf{S}^{(1,1)}_{u\cdot}\cdot\mathbf{R}_{\cdot i} = \sum_{v\in\mathcal{U}} \mathbf{S}^{(1,1)}_{uv}\cdot\mathbf{R}_{vi},$$

in which parameter $\mathbf{S}^{(1,1)}_{uv}$ is interpreted as the similarity between users u and v, i.e., $\mathbf{S}^{(1,1)}_{uv} = sim(u,v)$. $\mathbf{S}^{(1,1)}$ is often called the user-similarity matrix. The intuition behind this model is that good recommendations are preferred by similar users. Furthermore, Aiolli [2; 3] foresees the possibility to rescale the scores such that popular items get less importance. This changes the factorization model to

$$\mathbf{S} = \mathbf{S}^{(1,1)}\mathbf{R}\mathbf{S}^{(1,3)}, \qquad \mathbf{S}^{(1,3)}_{ji} = \mathbb{I}(j=i)\cdot c(i)^{-(1-\beta)},$$

in which $\mathbf{S}^{(1,3)}$ is a diagonal rescaling matrix that rescales the item scores according to the item popularity $c(i)$, and $\beta\in[0,1]$ a hyperparameter. Additionally, Aiolli [3] imposes the constraint that $\mathbf{S}^{(1,1)}$ needs to be row normalized:

$$||\mathbf{S}^{(1,1)}_{u\cdot}|| = 1.$$

To the best of our knowledge, there exists no user-based counterpart for the ItemRank model of Equation 12.

4.3.3 Explicitly Biased

Furthermore, it is, obviously, possible to add explicit bias terms to neighborhood models. Wang et al. [68] proposed an item-based neighborhood model with one bias term:

$$\mathbf{S} = \mathbf{S}^{(1,1)} + \mathbf{R}\mathbf{S}^{(2,2)}, \qquad \mathbf{S}^{(1,1)}_{ui} = b_i,$$

and an analogous user-based model:

$$\mathbf{S} = \mathbf{S}^{(1,1)} + \mathbf{S}^{(2,1)}\mathbf{R}, \qquad \mathbf{S}^{(1,1)}_{ui} = b_i.$$

4.3.4 Unified

Moreover, Verstrepen and Goethals [66] showed that the item- and user-based neighborhood models are two incomplete instances of the same general neighborhood model. Consequently they propose *KUNN*, a complete instance of this general neighborhood model. The model they propose can be written as a weighted sum of a user- and an item-based model, in which the weights depend on the user u and the item i for which a recommendation score is computed:

$$\begin{aligned}\mathbf{S} &= \left(\mathbf{S}^{(1,1)}\mathbf{R}\right)\mathbf{S}^{(1,3)} + \mathbf{S}^{(2,1)}\left(\mathbf{R}\mathbf{S}^{(2,3)}\right)\\ \mathbf{S}^{(1,3)}_{ij} &= \mathbb{I}(i=j)\cdot c(i)^{-1/2}\\ \mathbf{S}^{(2,1)}_{uv} &= \mathbb{I}(u=v)\cdot c(u)^{-1/2}.\end{aligned} \tag{14}$$

Finally, note that the above matrix factorization based descriptions of nearest neighbors methods imply that matrix factorization methods and neighborhood methods are not two separate approaches, but two perspectives on the same approach.

4.4 Factored Similarity Neighborhood Models

The item-similarity matrix $\mathbf{S}^{(1,2)}$ in Equation 10 contains $|\mathcal{I}|^2$ parameters, which is in practice often a very large number. Consequently, it can happen that the training data is not sufficient to accurately compute this many parameters. Furthermore, one often precomputes the item-similarity matrix $\mathbf{S}^{(1,2)}$ and performs the dotproducts $\mathbf{R}_{u\cdot}\cdot\mathbf{S}^{(1,2)}_{\cdot i}$ to compute the recommendation scores $\mathbf{S}_{ui}$ in real time. In this case, the dotproduct is between two $|\mathcal{I}|$-dimensional vectors, which is often prohibitive in real time, high traffic applications. One solution[2] is to factorize the similarity matrix, which leads to the following factorization model:

$$\mathbf{S} = \mathbf{R}\mathbf{S}^{(1,2)}\mathbf{S}^{(1,2)^T},$$

in which every row of $\mathbf{S}^{(1,2)} \in \mathbb{R}^{|\mathcal{I}|\times D}$ represents a D-dimensional item-profile vector, with D a hyperparameter [71; 8]. In this case the item-similarity matrix is equal to

$$\mathbf{S}^{(1,2)}\mathbf{S}^{(1,2)^T},$$

which means that the similarity $sim(i,j)$ between two items i and j is defined as the dotproduct of their respective profile vectors

$$\mathbf{S}^{(1,2)}_{i\cdot}\cdot\mathbf{S}^{(1,2)^T}_{j\cdot}.$$

This model only contains $|\mathcal{I}|\cdot D$ parameters instead of $|\mathcal{I}|^2$ which is much fewer since typically $D \ll |\mathcal{I}|$. Furthermore, by first precomputing the item vectors $\mathbf{S}^{(1,2)}$ and then precomputing the D-dimensional user-profile vectors given by $\mathbf{R}\mathbf{S}^{(1,2)}$, the real time computation of a score $\mathbf{S}_{ui}$ encompasses a dotproduct between a D-dimensional user-profile vector and a D-dimensional item-profile vector. Since $D \ll |\mathcal{I}|$, this dotproduct is much less expensive and can be more easily performed in real time. Furthermore, there is no trivial solution for the parameters of this model, as is the case for the non factored item-similarity model in Equation 10. Consequently, it is never required to impose the constraints from Equation 11. To avoid scaling problems when fitting parameters, Weston et al. [71] augment this model with a diagonal normalization factor matrix:

$$\mathbf{S} = \mathbf{S}^{(1,1)}\mathbf{R}\mathbf{S}^{(1,3)}\mathbf{S}^{(1,3)^T} \tag{15}$$

$$\mathbf{S}^{(1,1)}_{uv} = \mathbb{I}(u=v)\cdot c(u)^{-2/2}. \tag{16}$$

[2]Another solution is to enforce sparsity on the similarity matrix by means of the deviation function. This is discussed in Section 5.

A limitation of this model is that it implies that the similarity matrix is symmetric. This might hurt the model's accuracy in certain applications such as recommending tags for images. For an image of the Eiffel tower that is already tagged *Eiffel tower*, for example, the tag *Paris* is a reasonable recommendation. However, for an image of the Louvre already tagged *Paris*, the tag *Eiffel tower* is a bad recommendation. Paterek solved this problem for rating data by representing every item by two separate D-dimensional vectors [41]. One vector represents the item if it serves as evidence for computing recommendations, the other vector represents the item if it serves as a candidate recommendation. In this way, they can model also asymmetric similarities. This idea is not restricted to rating data, and for binary, positive-only data, it was adopted by Steck [58]:

$$\mathbf{S} = \mathbf{S}^{(1,1)}\mathbf{R}\mathbf{S}^{(1,3)}\mathbf{S}^{(1,4)} \tag{17}$$

$$\mathbf{S}^{(1,1)}_{uv} = \mathbb{I}(u=v)\cdot c(u)^{-1/2}. \tag{18}$$

Kabbur et al. also adopted this idea. However, similar to Equation 9, they also add bias terms:

$$\begin{aligned}\mathbf{S} &= \mathbf{S}^{(1,1)} + \mathbf{S}^{(2,1)} + \mathbf{S}^{(3,1)}\mathbf{R}\mathbf{S}^{(3,3)}\mathbf{S}^{(3,4)}\\ \mathbf{S}^{(1,1)}_{ui} &= b_u\\ \mathbf{S}^{(2,1)}_{ui} &= b_i\\ \mathbf{S}^{(3,1)}_{uv} &= \mathbb{I}(u=v)\cdot c(u)^{-\beta/2},\end{aligned} \tag{19}$$

with $\mathbf{S}^{(3,3)} \in \mathbb{R}^{|\mathcal{I}|\times D}$ the matrix of item profiles when they serve as evidence, $\mathbf{S}^{(3,4)} \in \mathbb{R}^{D\times|\mathcal{I}|}$ the matrix of item profiles when they serve as candidates , and $\beta \in [0,1]$ and D hyperparameters.

4.5 Higher Order Neighborhood Models

The nearest neighbors methods discussed up to this point only consider pairwise interactions $sim(j,i)$ and/or $sim(u,v)$ and aggregate all the relevant ones for computing recommendations. Several authors [10; 31; 64; 50; 30; 33; 7] have proposed to incorporate also higher order interactions $sim(J,i)$ and/or $sim(u,V)$ with $J \subset \mathcal{I}$ and $V \subset \mathcal{U}$. Also in this case we can distinguish item-based approaches from user-based approaches.

4.5.1 Item-based

For the item-based approach, most authors [10; 31; 64; 30; 7] propose to replace the user-item matrix $\mathbf{R}$ in the pairwise model of Equation 10 by the user-itemset matrix $\mathbf{X}$:

$$\begin{aligned}\mathbf{S} &= \mathbf{X}\mathbf{S}^{(1,2)}\\ \mathbf{X}_{uJ} &= \prod_{j\in J}\mathbf{R}_{uj},\end{aligned} \tag{20}$$

with $\mathbf{S}^{(1,2)} \in \mathbb{R}^{D\times|\mathcal{I}|}$ the itemset-item similarity matrix and $\mathbf{X} \in \{0,1\}^{|\mathcal{U}|\times D}$ the user-itemset matrix, in which $D \leq 2^{|\mathcal{I}|}$ is the result of an itemset selection procedure.

The HOSLIM method [7] adopts this model and additionally imposes the constraints in Equation 11.

The case in which $D = 2^{|\mathcal{I}|}$, and $\mathbf{S}^{(1,2)}$ is dense, is intractable. Tractable methods either limit $D \ll 2^{|\mathcal{I}|}$ or impose sparsity on $\mathbf{S}^{(1,2)}$ via the deviation function. While we discuss the latter in Section 5.11, there are multiple ways to do the former. Deshpande and Karypis [10] limit the number of itemsets by limiting the size of J. Alternatively, Christakopoulou and Karypis [7] only consider itemsets J that were preferred by more than a minimal number of users. van Leeuwen and Puspitaningrum, on the other hand, limit the number of higher order itemsets by using an itemset selection algorithm based on the minimal description length principle [64]. Finally, Menezes et al. claim that it is in certain applications possible to compute all higher order interactions if one computes all higher order interactions on demand instead of in advance [31]. However, delaying the computation does not reduce its exponential complexity. Only if a large portion of the users requires recommendations on a very infrequent basis, computations for these users can be spread over a very long period of time and their approach might be feasible.

An alternative model for incorporating higher order interactions between items consists of finding the best association rule for making recommendations [50; 33]. This corresponds to the matrix factorization model

$$\mathbf{S} = \mathbf{X} \otimes \mathbf{S}^{(1,2)}, \qquad \mathbf{X}_{uJ} = \prod_{j \in J} \mathbf{R}_{uj},$$

with

$$(\mathbf{A} \otimes \mathbf{B})_{xy} = \max_{i=1\ldots m} \mathbf{A}_{xi}\mathbf{B}_{iy},$$

in which $\mathbf{A} \in \mathbb{R}^{n \times m}$ and $\mathbf{B} \in \mathbb{R}^{m \times k}$.

We did not find a convincing motivation for either of the two aggregation strategies. Moreover, multiple authors report that their attempt to incorporate higher order interactions heavily increased computational costs and did not significantly improve the results [10; 64].

4.5.2 User-based

Incorporating higher order interactions between users can be achieved be replacing the user-item matrix in Equation 13 by the userset-item matrix $\mathbf{Y}$ [30; 60]:

$$\mathbf{S} = \mathbf{S}^{(1,1)}\mathbf{Y}, \qquad \mathbf{Y}_{Vi} = \prod_{v \in V} \mathbf{R}_{vi}, \tag{21}$$

with $\mathbf{Y} \in \{0,1\}^{D \times |\mathcal{I}|}$ and, $\mathbf{S}^{(1,1)} \in \mathbb{R}^{|\mathcal{U}| \times D}$ the user-userset similarity matrix, in which $D \leq 2^{|\mathcal{U}|}$ is the result of a userset selection procedure [30; 60].

4.6 Multi-profile Models

When multiple users, e.g., members of the same family, share a single account, or when a user has multiple distinct tastes, the above matrix factorization models can be too limited because they aggregate all the distinct tastes of an account into one vector [67; 70; 26]. Therefore, Weston et al. [70] propose MaxMF, in which they model every account with multiple vectors instead of just one. Then, for every candidate recommendation, their model chooses the vector that maximizes the score of the candidate:

$$\mathbf{S}_{ui} = \max_{p=1\ldots P} \left(\mathbf{S}^{(1,1,p)}\mathbf{S}^{(1,2)}\right)_{ui}.$$

Kabbur and Karypis [26] argue that this approach worsens the performance for accounts with homogeneous taste or a low number of known preferences and therefore propose, NLMFi, an adapted version that combines a global account profile with multiple taste-specific account profiles:

$$\mathbf{S}_{ui} = \left(\mathbf{S}^{(1,1)}\mathbf{S}^{(1,2)}\right)_{ui} + \max_{p=1\ldots P} \left(\mathbf{S}^{(2,1,p)}\mathbf{S}^{(2,2)}\right)_{ui}.$$

Alternatively, they also propose NLMFs, a version in which $\mathbf{S}^{(2,2)} = \mathbf{S}^{(1,2)}$, i.e., the item profiles are shared between the global and the taste-specific terms:

$$\mathbf{S} = \left(\mathbf{S}^{(1,1)}\mathbf{S}^{(1,2)}\right)_{ui} + \max_{p=1\ldots P} \left(\mathbf{S}^{(2,1,p)}\mathbf{S}^{(1,2)}\right)_{ui}.$$

An important downside of the above two models is that P, the number of distinct account-profiles, is a hyperparameter that is the same for every account and cannot be too large (typically 2 or 3) to avoid an explosion of the computational cost and the number of parameters. DAMIB-Cover [67], on the other hand, starts from the item-based model in Equation 10, and efficiently considers $P = 2^{c(u)}$ different profiles for every account u. Specifically, every profile corresponds to a subset $S^{(p)}$ of the known preferences of u, $I(u)$. This results in the factorization model

$$\mathbf{S}_{ui} = \max_{p=1\ldots 2^{c(u)}} \left(\mathbf{R}\mathbf{S}^{(1,2,p)}\mathbf{S}^{(1,3)}\right)_{ui}$$

$$\mathbf{S}^{(1,2,p)}_{jk} = \frac{\mathbb{I}(j=k) \cdot |S^{(p)} \cap \{j\}|}{|S^{(p)}|^{\beta}},$$

with $\mathbf{S}^{(1,2,p)}$ a diagonal matrix that selects and rescales the known preferences of u that correspond to the subset $S^{(p)} \subseteq I(u)$, and $\beta \in [0,1]$ a hyperparameter.

5. DEVIATION FUNCTIONS

The factorization models described in Sec. 4 contain many parameters, i.e., the entries in the a priori unknown factor matrices, which we collectively denote as θ. These parameters need to be computed based on the training data $\mathbf{R}$. Computing all the parameters in a factorization model is done by minimizing the *deviation* between the training data $\mathbf{R}$ and the parameters θ of the factorization model. This deviation is measured by a deviation function $\mathcal{D}(\theta, \mathbf{R})$. Many deviation functions exist, and every deviation function mathematically expresses a different interpretation of the concept *deviation.*

The majority of deviation functions is non-convex in the parameters θ. Consequently, minimization algorithms can only compute parameters that correspond to a local minimum. The initialization of the parameters in the factorization model and the chosen hyperparameters of the minimization algorithm determine which local minimum will be found. If the value of the deviation function in this local minimum is not much higher than that of the global minimum, it is considered a good minimum.

5.1 Probabilistic Scores-based

Hofmann [19] proposes to compute the *optimal* parameters θ^* of the model as those that maximize the loglikelihood of the model, i.e., $\log p(\mathbf{R}|\theta)$, the logprobability of the known preferences given the model:

$$\theta^* = \arg\max_{\theta} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} \mathbf{R}_{ui} \log p(\mathbf{R}_{ui}|\theta). \tag{22}$$

Furthermore, he models $p(\mathbf{R}_{ui} = 1|\theta)$ with $\mathbf{S}_{ui}$, i.e., he interprets the scores $\mathbf{S}$ as probabilities. This gives

$$\theta^* = \arg\max_{\theta} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} \mathbf{R}_{ui} \log \mathbf{S}_{ui},$$

which is equivalent to minimizing the deviation function

$$\mathcal{D}(\theta, \mathbf{R}) = -\sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} \mathbf{R}_{ui} \log \mathbf{S}_{ui}. \tag{23}$$

Recall that parameters θ are not directly visible in the right hand side of this equation but that scores $\mathbf{S}_{ui}$ are computed based on the factorization model which contains parameters θ, i.e., we use short notation $\mathbf{S}_{ui}$ instead of the full notation $\mathbf{S}_{ui}(\theta)$.

This deviation function was also adopted by Blei et al. in their approach called *latent dirichlet allocation (LDA)* [6].

Furthermore, note that Hofmann only maximizes the log-probability of the observed feedback and ignores the missing preferences. This is equivalent to the assumption that there is no information in the missing preferences, which implicitly corresponds to the assumption that feedback is *missing at random (MAR)* [56]. Clearly, this is not a realistic assumption, since negative feedback is missing by definition, which is obviously non random. Moreover, the number of items in collaborative filtering problems is typically very large, and only a very small subset of them will be preferred by a user. Consequently, the probability that a missing preference is actually not preferred is high. Hence, in reality, the feedback is *missing not at random (MNAR)*, and a good deviation function needs to account for this [56]. Furthermore, notice that Hofmann only maximizes the logprobability of the observed feedback and ignores the missing preferences. This is equivalent to the assumption that there is no information in the missing preferences, which implicitly corresponds to the assumption that feedback is *missing at random (MAR)* [56]. Clearly, this is not a realistic assumption, since negative feedback is missing by definition, which is obviously non random. Moreover, the number of items in collaborative filtering problems is typically very large, and only a very small subset of them will be preferred by a user. Consequently, the probability that a missing preference is actually not preferred is high. Hence, in reality, the feedback is *missing not at random (MNAR)*, and a good deviation function needs to account for this [56].

One approach is to assume, for the purposes of defining the deviation function, that *all* missing preferences are not preferred. This assumption is called *all missing are negative (AMAN)* [37]. Under this assumption, the parameters that maximize the loglikelihood of the model are computed as

$$\theta^* = \arg\max_{\theta} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} \log p(\mathbf{R}_{ui}|\theta).$$

For binary, positive-only data, one can model $p(\mathbf{R}_{ui}|\theta)$ as $\mathbf{S}_{ui}^{\mathbf{R}_{ui}} \cdot (1 - \mathbf{S}_{ui})^{(1-\mathbf{R}_{ui})}$. In this case, the parameters are computed as

$$\theta^* = \arg\max_{\theta} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} \left(\mathbf{R}_{ui} \log \mathbf{S}_{ui} + (1 - \mathbf{R}_{ui}) \log(1 - \mathbf{S}_{ui})\right).$$

While the AMAN assumption is more realistic than the MAR assumption, it adopts a conceptually flawed missing data model. Specifically, it assumes that all missing preferences are not preferred, which contradicts the goal of collaborative filtering: to find the missing preferences that are actually preferred. A better missing data model still assumes that all missing preferences are not preferred. However, it attaches a lower confidence to the assumption that a missing preference is not preferred, and a higher confidence to the assumption that an observed preference is indeed preferred. One possible way to apply this missing data model was proposed by Steck [56]. Although his original approach is more general, we give a specific simplified version that for binary, positive-only data:

$$\begin{aligned}\theta^* = \arg\max_{\theta} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} \big(&\mathbf{R}_{ui} \log \mathbf{S}_{ui} \\ &+ \alpha \cdot (1 - \mathbf{R}_{ui}) \log (1 - \mathbf{S}_{ui})\big),\end{aligned} \tag{24}$$

in which the hyperparameter $\alpha < 1$ attaches a lower importance to the contributions that correspond to $\mathbf{R}_{ui} = 0$. Johnson [24] proposed a very similar computation, but does not motivate why he deviates from the theoretically well founded version of Steck. Furthermore, Steck adds regularization terms to avoid overfitting and finally proposes the deviation function

$$\begin{aligned}\mathcal{D}(\theta, \mathbf{R}) = \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} \big(&-\mathbf{R}_{ui} \log \mathbf{S}_{ui} \\ &- \alpha \cdot (1 - \mathbf{R}_{ui}) \cdot \log(1 - \mathbf{S}_{ui}) \\ &+ \lambda \cdot \alpha \cdot (||\theta_u||_F^2 + ||\theta_i||_F^2)\big),\end{aligned}$$

with $|| \cdot ||_F$ the Frobenius norm, λ a regularization hyperparameter, and θ_u, θ_i the vectors that group the model parameters related to user u and item i, respectively.

5.2 Basic Squared Error-based

Most deviation functions, however, abandon the interpretation that the scores $\mathbf{S}$ are probabilities. In this case, one can choose to model $p(\mathbf{R}_{ui}|\theta)$ with a normal distribution $\mathcal{N}(\mathbf{R}_{ui}|\mathbf{S}_{ui}, \sigma)$. By additionally adopting the AMAN assumption, the *optimal* parameters are computed as the ones that maximize the loglikelihood $\log p(\mathbf{R}|\theta)$:

$$\theta^* = \arg\max_{\theta} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} \log \mathcal{N}(\mathbf{R}_{ui}|\mathbf{S}_{ui}, \sigma),$$

which is equivalent to

$$\begin{aligned}\theta^* &= \arg\max_{\theta} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} -(\mathbf{R}_{ui} - \mathbf{S}_{ui})^2 \\ &= \arg\min_{\theta} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} (\mathbf{R}_{ui} - \mathbf{S}_{ui})^2.\end{aligned}$$

The Eckart-Young theorem [13] states that the scores matrix $\mathbf{S}$ that results from these parameters θ^*, is the same as that found by singular value decomposition (SVD) with the same dimensions of the training data matrix $\mathbf{R}$. As such, the theorem relates the above approach of minimizing the squared error between $\mathbf{R}$ and $\mathbf{S}$ to *latent semantic analysis (LSA)* [9] and the SVD based collaborative filtering methods [8; 49].

Alternatively, it is possible to compute the *optimal* parameters as those that maximize the logposterior $\log p(\theta|\mathbf{R})$, which relates to the loglikelihood $\log p(\mathbf{R}|\theta)$ as

$$p(\theta|\mathbf{R}) \propto p(\mathbf{R}|\theta) \cdot p(\theta).$$

When $p(\theta)$, the prior distribution of the parameters, is chosen to be a zero-mean, spherical normal distribution, maximizing the logposterior is equivalent to minimizing the deviation function [32]

$$\mathcal{D}(\theta,\mathbf{R}) = \sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}\Big((\mathbf{R}_{ui}-\mathbf{S}_{ui})^2 + \lambda_u\cdot||\theta_u||_F^2 + \lambda_i\cdot||\theta_i||_F^2\Big),$$

with λ_u, λ_i regularization hyperparameters. Hence, maximizing the logposterior instead of the loglikelihood is equivalent to adding a regularization term. This deviation function is adopted by the FISM and NLMF methods [27; 26]. In an alternative interpretation of this deviation function, $\mathbf{S}$ is a factorized approximation of $\mathbf{R}$ and the deviation function minimizes the squared error of the approximation. The regularization with the Frobenius norm is added to avoid overfitting. For the SLIM and HOSLIM methods, an alternative regularization term is proposed:

$$\mathcal{D}(\theta,\mathbf{R}) = \sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}(\mathbf{R}_{ui}-\mathbf{S}_{ui})^2 + \sum_{t=1}^{T}\sum_{f=1}^{F}\lambda_R||\mathbf{S}^{(t,f)}||_F^2 + \lambda_1||\mathbf{S}^{(t,f)}||_1,$$

with $||\cdot||_1$ the l_1-norm. Whereas the role of the Frobenius-norm is to avoid overfitting, the role of the l_1-norm is to introduce sparsity. The combined use of both norms is called elastic net regularization, which is known to implicitly group correlated items [34]. The sparsity induced by the l_1-norm regularization lowers the memory required for storing the model and allows to speed-up the computation of recommendations by means of the sparse dotproduct. Even more sparsity can be obtained by fixing a majority of the parameters to 0, based on a simple feature selection method. Ning and Karypis [34] empirically show that this significantly reduces runtimes, while only slightly reducing the accuracy.

5.3 Weighted Squared Error-based

These squared error based deviation functions can also be adapted to diverge from the AMAN assumption to a missing data model that attaches lower confidence to the missing preferences [21; 37; 56]:

$$\mathcal{D}(\theta,\mathbf{R}) = \sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}\mathbf{W}_{ui}(\mathbf{R}_{ui}-\mathbf{S}_{ui})^2 + \sum_{t=1}^{T}\sum_{f=1}^{F}\lambda_{tf}||\mathbf{S}^{(t,f)}||_F^2, \quad (25)$$

in which $\mathbf{W}\in\mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ assigns weights to values in $\mathbf{R}$. The higher $\mathbf{W}_{ui}$, the higher the confidence about $\mathbf{R}_{ui}$. Hu et al. [21] provide two potential definitions of $\mathbf{W}$:

$$\mathbf{W}_{ui} = 1 + \beta\mathbf{R}_{ui},$$
$$\mathbf{W}_{ui} = 1 + \alpha\log(1+\mathbf{R}_{ui}/\epsilon),$$

with α, β, ϵ hyperparameters. Clearly, this method is not limited to binary data, but works on positive-only data in general. We, however, only consider its use for binary, positive-only data. Equivalently, Steck [56] proposed:

$$\mathbf{W}_{ui} = \mathbf{R}_{ui} + (1-\mathbf{R}_{ui})\cdot\alpha,$$

with $\alpha < 1$.
Additionally, Steck [57] pointed out that a preference is more likely to show up in the training data if the item is more popular. To compensate for this bias, Steck proposes to weight the known preferences non-uniformly:

$$\mathbf{W}_{ui} = \mathbf{R}_{ui}\cdot\frac{C}{c(i)^\beta} + (1-\mathbf{R}_{ui})\cdot\alpha,$$

with C a constant, R the number of non-zeros in the training data $\mathbf{R}$, and $\beta\in[0,1]$ a hyperparameter. Analogously, a preference is more likely to be missing in the training data if the item is less popular. To compensate for this bias, Steck proposes to weight the missing preferences non-uniformly:

$$\mathbf{W}_{ui} = \mathbf{R}_{ui} + (1-\mathbf{R}_{ui})\cdot C\cdot c(i)^\beta, \quad (26)$$

with C a constant. Steck proposed these two weighting strategies as alternatives to each other. However, we believe that they can be combined since they are the application of the same idea to the known and missing preferences respectively.
Although they provide less motivation, Pan et al. [37] arrive at similar weighting schemes. They propose $\mathbf{W}_{ui} = 1$ if $\mathbf{R}_{ui} = 1$ and give three possibilities for the case when $\mathbf{R}_{ui} = 0$:

$$\mathbf{W}_{ui} = \delta, \quad (27)$$
$$\mathbf{W}_{ui} = \alpha\cdot c(u), \quad (28)$$
$$\mathbf{W}_{ui} = \alpha\left(|\mathcal{U}| - c(i)\right), \quad (29)$$

with $\delta\in[0,1]$ a uniform hyperparameter and α a hyperparameter such that $\mathbf{W}_{ui}\le 1$ for all pairs (u,i) for which $\mathbf{R}_{ui} = 0$. In the first case, all missing preferences get the same weight. In the second case, a missing preference is more negative if the user already has many preferences. In the third case, a missing preference is less negative if the item is popular. Interestingly, the third weighting scheme is orthogonal to the one of Steck in Equation 26. Additionally, Pan et al. [37] propose a deviation function with an alternative regularization:

$$\mathcal{D}(\theta,\mathbf{R}) = \sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}\mathbf{W}_{ui}\Big((\mathbf{R}_{ui}-\mathbf{S}_{ui})^2 + \lambda\left(||\theta_u||_F^2 + ||\theta_i||_F^2\right)\Big). \quad (30)$$

Yao et al. [73] adopt a more complex missing data model that has a hyperparameter p, which indicates the overall likelihood that a missing preference is preferred. This translates into the deviation function:

$$\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}\mathbf{R}_{ui}\mathbf{W}_{ui}(1-\mathbf{S}_{ui})^2 + \sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}(1-\mathbf{R}_{ui})\mathbf{W}_{ui}(p-\mathbf{S}_{ui})^2 + \sum_{t=1}^{T}\sum_{f=1}^{F}\lambda_{tf}||\mathbf{S}^{(t,f)}||_F^2 \quad (31)$$

The special case with $p = 0$ reduces this deviation function to the one in Equation 25.
An even more complex missing data model and correspond-

ing deviation function was proposed by Sindhwani et al. [55]:

$$\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}\mathbf{R}_{ui}\mathbf{W}_{ui}(1-\mathbf{S}_{ui})^2 + \sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}(1-\mathbf{R}_{ui})\mathbf{W}_{ui}\left(\mathbf{P}_{ui}-\mathbf{S}_{ui}\right)^2 + \sum_{t=1}^{T}\sum_{f=1}^{F}\lambda_{tf}||\mathbf{S}^{(t,f)}||_F^2 - \lambda_H\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}(1-\mathbf{R}_{ui})H(\mathbf{P}_{ui}) \quad (32)$$

with $\mathbf{P}$ and $\mathbf{W}$ additional parameters of the model that need to be computed based on the data $\mathbf{R}$, together with all other parameters. The last term contains the entropy function H and serves as a regularizer for $\mathbf{P}$. Furthermore, they define the constraint

$$\frac{1}{|\mathcal{U}||\mathcal{I}|-|\mathcal{R}|}\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}\mathbf{P}_{ui}=p,$$

which expresses that the average probability that a missing value is actually one must be equal to the hyperparameter p. To reduce the computational cost, they fix $\mathbf{P}_{ui}=0$ for most (u,i)-pairs and randomly choose a few (u,i)-pairs for which $\mathbf{P}_{ui}$ is computed based on the data. It seems that this simplification completely offsets the modeling flexibility that was obtained by introducing $\mathbf{P}$. Additionally, they simplify $\mathbf{W}$ as the one-dimensional matrix factorization

$$\mathbf{W}_{ui}=\mathbf{V}_u\mathbf{V}_i.$$

A conceptual inconsistency of this deviation function is that although the recommendation score is given by $\mathbf{S}_{ui}$, $\mathbf{P}_{ui}$ could also be used. Hence, there exist two parameters for the same concept, which is, at best, ambiguous.

5.4 Maximum Margin-based

A disadvantage of the squared error-based deviation functions is their symmetry. For example, if $\mathbf{R}_{ui}=1$ and $\mathbf{S}_{ui}=0$, $(\mathbf{R}_{ui}-\mathbf{S}_{ui})^2=1$. This is desirable behavior because we want to penalize the model for predicting that a preference is not preferred. However if $\mathbf{S}_{ui}=2$, we obtain the same penalty: $(\mathbf{R}_{ui}-\mathbf{S}_{ui})^2=1$. This, on the other hand, is not desirable because we do not want to penalize the model for predicting that a preference will definitely be preferred.

A maximum margin-based deviation function does not suffer from this problem [36]:

$$\mathcal{D}\left(\theta,\mathbf{R}\right)=\sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}}\mathbf{W}_{ui}\cdot h\left(\tilde{\mathbf{R}}_{ui}\cdot\mathbf{S}_{ui}\right)+\lambda||\mathbf{S}||_{\Sigma}, \quad (33)$$

with $||.||_{\Sigma}$ the trace norm, λ a regularization hyperparameter, $h\left(\tilde{\mathbf{R}}_{ui}\cdot\mathbf{S}_{ui}\right)$ a smooth hinge loss given by Figure 3 [46], $\mathbf{W}$ given by one of the Equations 27-29 and the matrix $\tilde{\mathbf{R}}$ defined as

$$\begin{cases}\tilde{\mathbf{R}}_{ui}=1 & \textit{if } \mathbf{R}_{ui}=1\\ \tilde{\mathbf{R}}_{ui}=-1 & \textit{if } \mathbf{R}_{ui}=0.\end{cases}$$

This deviation function incorporates the confidence about the training data by means of $\mathbf{W}$ and the missing knowledge about the degree of preference by means of the hinge loss $h\left(\tilde{\mathbf{R}}_{ui}\cdot\mathbf{S}_{ui}\right)$. Since the degree of a preference $\mathbf{R}_{ui}=1$ is considered unknown, a value $\mathbf{S}_{ui}>1$ is not penalized if $\mathbf{R}_{ui}=1$.

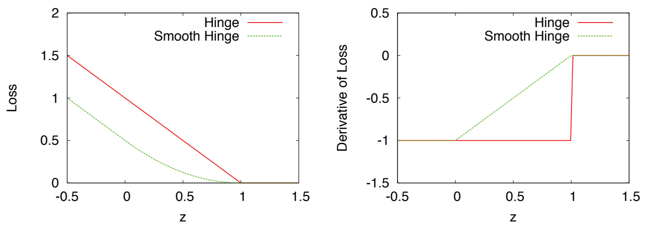

Figure 3: Shown are the loss function values $h(z)$ (left) and the gradients $dh(z)/dz$ (right) for the Hinge and Smooth Hinge. Note that the gradients are identical outside the region $z\in(0,1)$ [46].

5.5 Overall Ranking Error-based

The scores $\mathbf{S}$ computed by a collaborative filtering method are used to personally rank all items for every user. Therefore, one can argue that it is more natural to directly optimize the ranking of the items instead of their individual scores.

Rendle et al. [45], aim to maximize the area under the ROC curve (AUC), which is given by:

$$AUC=\frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}\frac{1}{|u|\cdot(|\mathcal{I}|-|u|)}\cdot\sum_{\mathbf{R}_{ui}=1}\sum_{\mathbf{R}_{uj}=0}\mathbb{I}(\mathbf{S}_{ui}>\mathbf{S}_{uj}). \quad (34)$$

If the AUC is higher, the pairwise rankings induced by the model $\mathbf{S}$ are more in line with the observed data $\mathbf{R}$. However, because $\mathbb{I}(\mathbf{S}_{ui}>\mathbf{S}_{uj})$ is non-differentiable, it is impossible to actually compute the parameters that (locally) maximize the AUC. Their solution is a deviation function, called the *Bayesian Personalized Ranking(BPR)-criterium*, which is a differentiable approximation of the negative AUC from which constant factors have been removed and to which a regularization term has been added:

$$\mathcal{D}\left(\theta,\mathbf{R}\right)=\sum_{u\in\mathcal{U}}\sum_{\mathbf{R}_{ui}=1}\sum_{\mathbf{R}_{uj}=0}-\log\sigma(\mathbf{S}_{ui}-\mathbf{S}_{uj}) + \sum_{t=1}^{T}\sum_{f=1}^{F}\lambda_{tf}||\mathbf{S}^{(t,f)}||_F^2, \quad (35)$$

with $\sigma(\cdot)$ the sigmoid function and λ_{tf} regularization hyperparameters. Since this approach explicitly accounts for the missing data, it corresponds to the MNAR assumption.

Pan and Chen claim that it is beneficial to relax the BPR deviation function by Rendle et al. to account for noise in the data [38; 39]. Specifically, they propose CoFiSet [38], which allows certain violations $\mathbf{S}_{ui}<\mathbf{S}_{uj}$ when $\mathbf{R}_{ui}>\mathbf{R}_{uj}$:

$$\mathcal{D}\left(\theta,\mathbf{R}\right)=\sum_{u\in\mathcal{U}}\sum_{I\subseteq I(u)}\sum_{J\subseteq\mathcal{I}\setminus I(u)} -\log\sigma\left(\frac{\sum_{i\in I}\mathbf{S}_{ui}}{|I|}-\frac{\sum_{j\in J}\mathbf{S}_{uj}}{|J|}\right) +\Gamma(\theta), \quad (36)$$

with $\Gamma(\theta)$ a regularization term that slightly deviates from the one proposed by Rendle et al. for no clear reason. Alternatively, they propose GBPR [39], which relaxes the BPR deviation function in a different way:

$$\mathcal{D}(\theta, \mathbf{R}) = \sum_{u \in \mathcal{U}} \sum_{\mathbf{R}_{ui}=1} \sum_{\mathbf{R}_{uj}=0} \Gamma(\theta) - \log \sigma^2 \left(\alpha \cdot \frac{\sum_{g \in \mathcal{G}_{u,i}} \mathbf{S}_{gi}}{|\mathcal{G}_{u,i}|} + (1-\alpha) \cdot \mathbf{S}_{ui} - \mathbf{S}_{uj} \right), \quad (37)$$

with $\mathcal{G}_{u,i}$ the union of $\{u\}$ with a random subset of $\{g \in \mathcal{U} \setminus \{u\} | \mathbf{R}_{gi} = 1\}$, and α a hyperparameter.
Furthermore, Kabbur et al. [27] also aim to maximize AUC with their method FISMauc. However, they propose to use a different differentiable approximation of AUC:

$$\mathcal{D}(\theta, \mathbf{R}) = \sum_{u \in \mathcal{U}} \sum_{\mathbf{R}_{ui}=1} \sum_{\mathbf{R}_{uj}=0} \left((\mathbf{R}_{ui} - \mathbf{R}_{uj}) - (\mathbf{S}_{ui} - \mathbf{S}_{uj}) \right)^2 + \sum_{t=1}^{T} \sum_{f=1}^{F} \lambda_{tf} ||\mathbf{S}^{(t,f)}||_F^2. \quad (38)$$

The same model without regularization was proposed by Töscher and Jahrer [62]:

$$\mathcal{D}(\theta, \mathbf{R}) = \sum_{u \in \mathcal{U}} \sum_{\mathbf{R}_{ui}=1} \sum_{\mathbf{R}_{uj}=0} \left((\mathbf{R}_{ui} - \mathbf{R}_{uj}) - (\mathbf{S}_{ui} - \mathbf{S}_{uj}) \right)^2. \quad (39)$$

A similar deviation function was proposed by Takács and Tikk [61]:

$$\mathcal{D}(\theta, \mathbf{R}) = \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} \mathbf{R}_{ui} \sum_{j \in \mathcal{I}} w(j) \cdot \left((\mathbf{S}_{ui} - \mathbf{S}_{uj}) - (\mathbf{R}_{ui} - \mathbf{R}_{uj}) \right)^2, \quad (40)$$

with $w(\cdot)$ a user-defined item weighting function. The simplest choice is $w(j) = 1$ for all j. An alternative proposed by Takács and Tikk is $w(j) = c(j)$. Another important difference is that this deviation function also minimizes the score-difference between the known preferences mutually.
Finally, it is remarkable that both Töscher and Jahrer, and Takács and Tikk, explicitly do not add a regularization term, whereas most other authors find that the regularization term is important for their model's performance.

5.6 Top of Ranking Error-based

Very often, only the N highest ranked items are shown to users. Therefore, Shi et al. [52] propose to improve the top of the ranking at the cost of worsening the overall ranking. Specifically, they propose the CLiMF method, which aims to minimize the *mean reciprocal rank (MRR)* instead of the AUC. The MRR is defined as

$$MRR = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} r_{>} \left(\max_{\mathbf{R}_{ui}=1} \mathbf{S}_{ui} \mid \mathbf{S}_{u\cdot} \right)^{-1},$$

in which $r_{>}(a|\mathbf{b})$ gives the rank of a among all numbers in $\mathbf{b}$ when ordered in descending order. Unfortunately, the non-smoothness of $r_{>}()$ and max makes the direct optimization of MRR unfeasible. Hence, Shi et al. derive a differentiable version of MRR. Yet, optimizing it is intractable. Therefore, they optimize a lower bound instead. After also adding regularization terms, their final deviation function is given by

$$\mathcal{D}(\theta, \mathbf{R}) = -\sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} \mathbf{R}_{ui} \Big(\log \sigma(\mathbf{S}_{ui}) + \sum_{j \in \mathcal{I}} \log\left(1 - \mathbf{R}_{uj} \sigma(\mathbf{S}_{uj} - \mathbf{S}_{ui})\right) \Big) + \sum_{t=1}^{T} \sum_{f=1}^{F} \lambda_{tf} ||\mathbf{S}^{(t,f)}||_F^2 \quad (41)$$

with λ a regularization constant and $\sigma(\cdot)$ the sigmoid function. A disadvantage of this deviation function is that it ignores all missing data, i.e., it corresponds to the MAR assumption.
An alternative for MRR is *mean average precision (MAP)*:

$$MAP = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{c(u)} \sum_{\mathbf{R})_{ui}=1} \frac{1}{r_{>}(\mathbf{S}_{ui} | \mathbf{S}_{u\cdot})} \sum_{\mathbf{R}_{uj}=1} \mathbb{I}(\mathbf{S}_{uj} \geq \mathbf{S}_{ui}),$$

which still emphasizes the top of the ranking, but less extremely than MRR. However, MAP is also non-smooth, preventing its direct optimization. For MAP, too, Shi et al. derived a differentiable version, called TFMAP [51]:

$$\mathcal{D}(\theta, \mathbf{R}) = -\sum_{u \in \mathcal{U}} \frac{1}{c(u)} \sum_{\mathbf{R})_{ui}=1} \sigma(\mathbf{S}_{ui}) \sum_{\mathbf{R}_{uj}=1} \sigma(\mathbf{S}_{uj} - \mathbf{S}_{ui}) + \lambda \cdot \sum_{t=1}^{T} \sum_{f=1}^{F} ||\mathbf{S}^{(t,f)}||_F^2, \quad (42)$$

with λ a regularization hyperparameter. Besides, the formulation of TFMAP in Equation 42 is a simplified version of the original, concieved for multi-context data, which is out of the scope of this survey.
Dhanjal et al. proposed yet another alternative [12]. They start from the definition of AUC in Equation 34, approximate the indicator function $\mathbb{I}(\mathbf{S}_{ui} - \mathbf{S}_{uj})$ by the squared hinge loss $(\max(0, 1 + \mathbf{S}_{uj} - \mathbf{S}_{ui}))^2$ and emphasize the deviation at the top of the ranking by means of the hyperbolic tangent function $\tanh(\cdot)$:

$$\mathcal{D}(\theta, \mathbf{R}) = \sum_{u \in \mathcal{U}} \sum_{\mathbf{R}_{ui}=1} \tanh\left(\sum_{\mathbf{R}_{u}j=0} (\max(0, 1 + \mathbf{S}_{uj} - \mathbf{S}_{ui}))^2 \right). \quad (43)$$

5.7 k-Order Statistic-based

On the one hand, the deviation functions in Equations 35-40 try to minimize the overall rank of the known preferences. On the other hand, the deviation functions in Equations 41 and 43 try to push one or a few known preferences as high as possible to the top of the item-ranking. Weston et al. [71] propose to minimize a trade-off between the above two extremes:

$$\sum_{u\in\mathcal{U}}\sum_{\mathbf{R}_{ui}=1} w\left(\frac{r_>(\mathbf{S}_{ui}\mid\{\mathbf{S}_{ui}\mid\mathbf{R}_{ui}=1\})}{c(u)}\right) \cdot \sum_{\mathbf{R}_{uj}=0}\frac{\mathbb{I}(\mathbf{S}_{uj}+1\geq\mathbf{S}_{ui})}{r_>(\mathbf{S}_{uj}\mid\{\mathbf{S}_{uk}\mid\mathbf{R}_{uk}=0\})}, \quad (44)$$

with $w(\cdot)$ a function that weights the importance of the known preference as a function of their predicted rank among all known preferences. This weighting function is user-defined and determines the trade-off between the two extremes, i.e., minimizing the mean rank of the known preferences and minimizing the maximal rank of the known preferences. Because this deviation function is non-differentiable, Weston et al. propose the differentiable approximation

$$\mathcal{D}(\theta,\mathbf{R})=\sum_{u\in\mathcal{U}}\sum_{\mathbf{R}_{ui}=1} w\left(\frac{r_>(\mathbf{S}_{ui}\mid\{\mathbf{S}_{ui}\mid\mathbf{R}_{ui}=1\})}{c(u)}\right) \cdot \sum_{\mathbf{R}_{uj}=0}\frac{\max(0,1+\mathbf{S}_{uj}-\mathbf{S}_{ui})}{N^{-1}(|\mathcal{I}|-c(u))}, \quad (45)$$

where they replaced the indicator function by the hinge-loss and approximated the rank with $N^{-1}(|\mathcal{I}|-c(u))$, in which N is the number of items k that were randomly sampled until $\mathbf{S}_{uk}+1>\mathbf{S}_{ui}$[3]. Furthermore, they use the simple weighting function

$$w\left(\frac{r_>(\mathbf{S}_{ui}\mid\{\mathbf{S}_{ui}\mid\mathbf{R}_{ui}=1\})}{|u|}\right)= \begin{cases}1 \textbf{ if } r_>(\mathbf{S}_{ui}\mid S\subseteq\{\mathbf{S}_{ui}\mid\mathbf{R}_{ui}=1\},|S|=K)=k \textbf{ and}\\ 0 \textbf{ otherwise}\end{cases},$$

i.e., from the set S that contains K randomly sampled known preferences, ranked by their predicted score, only the item at rank k is selected to contribute to the training error. When k is set low, the top of the ranking will be optimized at the cost of a worse mean rank. The opposite will hold when k is set high. The regularization is not done by adding a regularization term but by forcing the norm of the factor matrices to be below a maximum, which is a hyperparameter.

Alternatively, Weston et al. also propose a simplified deviation function:

$$\mathcal{D}(\theta,\mathbf{R})=\sum_{u\in\mathcal{U}}\sum_{\mathbf{R}_{ui}=1} w\left(\frac{r_>(\mathbf{S}_{ui}\mid\{\mathbf{S}_{ui}\mid\mathbf{R}_{ui}=1\})}{c(u)}\right) \cdot \sum_{\mathbf{R}_{uj}=0}\max(0,1+\mathbf{S}_{uj}-\mathbf{S}_{ui}). \quad (46)$$

5.8 Rank Link Function-based

The ranking-based deviation functions discussed so far, are all tailor made differentiable approximations with respect to the recommendation scores of a certain ranking quality measure, like AUC, MRR or the k-th order statistic. Steck [58] proposes a more general approach that is applicable to any ranking quality measure that is differentiable with respect to the rankings. He demonstrates his method on two ranking quality measures: AUC and $nDCG$. For AUC_u, the AUC for user u, he does not use the formulation in Equation 34, but uses the equivalent

$$AUC_u=\frac{1}{c(u)\cdot(|\mathcal{I}|-c(u))} \cdot\left[(|\mathcal{I}|+1)c(u)-\binom{c(u)+1}{2}-\sum_{\mathbf{R}_ui=1}r_{ui}\right],$$

with r_{ui} the rank of item i in the recommendation list of user u. Second, $nDCG_u$ is defined as

$$nDCG_u=\frac{DCG}{DCG_{opt}},\qquad DCG=\sum_{\mathbf{R}_{ui}=1}\frac{1}{\log(r_{ui}+1)},$$

with DCG_{opt} the DCG of the optimal ranking. In both cases, Steck proposes to relate the rank r_{ui} with the recommendation score $\mathbf{R}_{ui}$ by means of a link function

$$r_{ui}=\max\left\{1,|\mathcal{I}|\cdot\left(1-cdf(\hat{\mathbf{S}}_{ui})\right)\right\}, \quad (47)$$

with $\hat{\mathbf{S}}_{ui}=(\mathbf{S}_{ui}-\mu_u)/std_u$ the normalized recommendation score in which μ_u and std_u are the mean and standard deviation of the recommendation scores for user u, and cdf is the cumulative distribution of the normalized scores. However, to know cdf, he needs to assume a distribution for the normalized recommendation scores. He motivates that a zero-mean normal distribution of the recommendation scores is a reasonable assumption. Consequently, $cdf(\hat{\mathbf{S}}_{ui})=probit(\hat{\mathbf{S}}_{ui})$. Furthermore, he proposes to approximate the probit function with the computationally more efficient sigmoid function or the even more efficient function

$$rankSE(\hat{\mathbf{S}}_{ui})=[1-([1-\hat{\mathbf{S}}_{ui}]_+)^2]_+,$$

with $[x]_+=max\{0,x\}$. In his *pointwise* approach, Steck uses Equation 47 to compute the ranks based on the recommendation scores. In his *listwise* approach, on the other hand, he finds the actual rank of a recommendation score by sorting the recommendation scores for every user, and uses Equation 47 only to compute the gradient of the rank with respect to the recommendation score during the minimization procedure. After adding regularizaton terms, he proposes the deviation function

$$\mathcal{D}(\theta,\mathbf{R})=\sum_{u\in\mathcal{U}}\Bigg(\sum_{\mathbf{R}_{ui}=1}\left(L(\mathbf{S}_{ui})+\lambda\cdot\left(||\theta_u||_F^2+||\theta_i||_F^2\right)\right) +\sum_{j\in\mathcal{I}}\gamma\cdot[\mathbf{S}_{uj}]_+^2\Bigg), \quad (48)$$

with θ_u,θ_i the vectors that group all model parameters related to user u and item i, respectively, λ,γ regularization hyperparameters, and $L(\mathbf{S}_{ui})$ equal to $-AUC_u$ or $-nDCG_u$, which are a function of $\mathbf{S}_{ui}$ via r_{ui}. The last regularization term is introduced to enforce an approximately normal distribution on the scores.

5.9 Posterior KL-divergence-based

In our framework, the *optimal* parameters θ^* are computed as $\theta^*=\arg\min_\theta\mathcal{D}(\theta,\mathbf{R})$. However, we can consider this a special case of

$$\theta^*=\psi\left(\arg\min_\phi\mathcal{D}(\theta(\phi),\mathbf{R})\right),$$

[3]Weston et al. [69] provide a justification for this approximation.

in which ψ is chosen to be the identity function and the parameters θ are identical to the parameters ϕ, i.e., $\theta(\phi) = \phi$. Now, some authors [28; 40; 16] propose to choose $\psi()$ and ϕ differently.

Specifically, they model every parameter θ_j of the factorization model as a random variable with a parameterized posterior probability density function $p(\theta_j|\phi_j)$. Hence, finding the variables ϕ corresponds to finding the posterior distributions of the model parameters θ. Because it is intractable to find the true posterior distribution $p(\theta|\mathbf{R})$ of the parameters, they settle for a *mean-field* approximation $q(\theta|\phi)$, in which all variables are assumed to be independent.

Then, they define ψ as $\psi(\phi^*) = \mathbb{E}_{q(\theta|\phi^*)}[\theta]$, i.e., the point-estimate of the parameters θ equals their expected value under the mean-field approximation of their posterior distributions. Note that $\theta^*_j = \mathbb{E}_{q(\theta_j|\phi^*_j)}[\theta_j]$ because of the independence assumption.

If all parameters θ_j are assumed to be normally distributed with mean μ_j and variance σ^2_j [28; 40], $\phi_j = (\mu_j, \sigma_j)$ and $\theta^*_j = \mathbb{E}_{q(\theta_j|\phi^*_j)}[\theta_j] = \mu^*_j$. If, on the other hand, all parameters θ_j are assumed to be gamma distributed with shape α_j and rate β_j [16], $\phi_j = (\alpha_j, \beta_j)$ and $\theta^*_j = \mathbb{E}_{q(\theta_j|\phi^*_j)}[\theta_j] = \alpha^*_j/\beta^*_j$.

Furthermore, prior distributions are defined for all parameters θ. Typically, when this approach is adopted, the underlying assumptions are represented as a *graphical model* [5].

The parameters ϕ, and therefore the corresponding mean-field approximations of the posterior distribution of the parameters θ, can be inferred by defining the deviation function as the KL-divergence of the real (unknown) posterior distribution of the parameters, $p(\theta|\mathbf{R})$, from the modeled posterior distribution of the parameters, $q(\theta|\phi)$,

$$\mathcal{D}\left(\theta(\phi), \mathbf{R}\right) = D_{KL}\left(q(\theta|\phi) \| p(\theta|\mathbf{R})\right),$$

which can be solved despite the fact that $p(\theta|\mathbf{R})$ is unknown [25]. This approach goes by the name *variational inference* [25].

A *nonparametric* version of this approach also considers D, the number of latent dimensions in the simple two factor factorization model of Equation 7, as a parameter that depends on the data $\mathbf{R}$ instead of a hyperparameter, as most other methods do [17].

Note that certain solutions for latent Dirichlet allocation [6] also use variational inference techniques. However, in this case, variational inference is a part of the (variational) expectation-maximization algorithm for computing the parameters that optimize the negative log-likelihood of the model parameters, which serves as the deviation function. This is different from the methods discussed in this section, where the KL-divergence between the real and the approximate posterior is the one and only deviation function.

5.10 Convex

An intuitively appealing but non-convex deviation function is worthless if there exists no algorithm that can efficiently compute parameters that correspond to its good local minimum. Convex deviation functions on the other hand, have only one global minimum that can be computed with one of the well studied convex optimization algorithms. For this reason, it is worthwhile to pursue convex deviation functions.

Aiolli [3] proposes a convex deviation function based on the AUC (Eq. 34). For every individual user $u \in \mathcal{U}$, Aiolli starts from AUC_u, the AUC for u:

$$AUC_u = \frac{1}{c(u) \cdot (|\mathcal{I}| - c(u))} \sum_{\mathbf{R}_{ui}=1} \sum_{\mathbf{R}_{uj}=0} \mathbb{I}(\mathbf{S}_{ui} > \mathbf{S}_{uj}).$$

Next, he proposes a lower bound on AUC_u:

$$AUC_u \geq \frac{1}{c(u) \cdot (|\mathcal{I}| - c(u))} \sum_{\mathbf{R}_{ui}=1} \sum_{\mathbf{R}_{uj}=0} \frac{\mathbf{S}_{ui} - \mathbf{S}_{uj}}{2},$$

and interprets it as a weighted sum of margins $\frac{\mathbf{S}_{ui} - \mathbf{S}_{uj}}{2}$ between any known preferences and any absent feedback, in which every margin gets the same weight $\frac{1}{c(u) \cdot (|\mathcal{I}| - c(u))}$. Hence maximizing this lower bound on the AUC corresponds to maximizing the sum of margins between any known preference and any absent feedback in which every margin has the same weight. A problem with maximizing this sum is that very high margins on pairs that are easily ranked correctly can hide poor (negative) margins on pairs that are difficult to rank correctly. Aiolli proposes to replace the uniform weights with a weighting scheme that emphasizes the difficult pairs such that the total margin is the worst possible case, i.e., the lowest possible sum of weighted margins. Specifically, he proposes to solve for every user u the joint optimization problem

$$\theta^* = \arg\max_{\theta} \min_{\alpha_{u*}} \sum_{\mathbf{R}_{ui}=1} \sum_{\mathbf{R}_{uj}=0} \alpha_{ui}\alpha_{uj}(\mathbf{S}_{ui} - \mathbf{S}_{uj}),$$

where for every user u, it holds that $\sum_{\mathbf{R}_{ui}=1} \alpha_{ui} = 1$ and $\sum_{\mathbf{R}_{uj}=0} \alpha_{uj} = 1$. To avoid overfitting of α, he adds two regularization terms:

$$\theta^* = \arg\max_{\theta} \min_{\alpha_{u*}} \Bigg(\sum_{\mathbf{R}_{ui}=1} \sum_{\mathbf{R}_{uj}=0} \alpha_{ui}\alpha_{uj}(\mathbf{S}_{ui} - \mathbf{S}_{uj}) + \lambda_p \sum_{\mathbf{R}_{ui}=1} \alpha^2_{ui} + \lambda_n \sum_{\mathbf{R}_{ui}=0} \alpha^2_{ui} \Bigg),$$

with λ_p, λ_n regularization hyperparameters. The model parameters θ, on the other hand, are regularized by normalization constraints on the factor matrices. Solving the above maximization for every user, is equivalent to minimizing the deviation function

$$\mathcal{D}\left(\theta, \mathbf{R}\right) = \sum_{u \in \mathcal{U}} \Bigg(\max_{\alpha_{u*}} \Bigg(\sum_{\mathbf{R}_{ui}=1} \sum_{\mathbf{R}_{uj}=0} \alpha_{ui}\alpha_{uj}(\mathbf{S}_{uj} - \mathbf{S}_{ui}) - \lambda_p \sum_{\mathbf{R}_{ui}=1} \alpha^2_{ui} - \lambda_n \sum_{\mathbf{R}_{ui}=0} \alpha^2_{ui} \Bigg) \Bigg). \quad (49)$$

5.11 Analytically Solvable

Some deviation functions are not only convex, but also analytically solvable. This means that the parameters that minimize these deviation functions can be exactly computed from a formula and that no numerical optimization algorithm is required.

Traditionally, methods that adopt these deviation functions have been inappropriately called *neighborhood* or *memory-based* methods. First, although these methods adopt neighborhood-based factorization models, there are also neighborhood-based methods that adopt non-convex deviation functions, such as SLIM [34] and BPR-kNN [45], which were, amongst others, discussed in Section 4. Second, a *memory-based* implementation of these methods, in which the necessary parameters of the factorization model are not precomputed, but computed in real time when they are required is conceptually possible, yet practically intractable in most cases. Instead, a *model-based* implementation of these methods, in which the factorization model is precomputed, is the best choice for the majority of applications.

5.11.1 Basic Neighborhood-based

A first set of analytically solvable deviation functions is tailored to the item-based neighborhood factorization models of Equation 10:

$$\mathbf{S} = \mathbf{R}\mathbf{S}^{(1,2)}.$$

As explained in Section 4, the factor matrix $\mathbf{S}^{(1,2)}$ can be interpreted as an item-similarity matrix. Consequently, these deviation functions compute every parameter in $\mathbf{S}^{(1,2)}$ as

$$\mathbf{S}^{(1,2)}_{ji} = \mathit{sim}(j,i),$$

with $\mathit{sim}(j,i)$ the similarity between items j and i according to some analytically computable similarity function. This is equivalent to

$$\mathbf{S}^{(1,2)}_{ji} - \mathit{sim}(j,i) = 0,$$

which is true for all (j,i)-pairs if and only if

$$\sum_{j\in\mathcal{I}}\sum_{i\in\mathcal{I}}\left(\mathbf{S}^{(1,2)}_{ji} - \mathit{sim}(j,i)\right)^2 = 0.$$

Hence, computing the factor matrix $\mathbf{S}^{(1,2)}_{ji}$ corresponds to minimizing the deviation function

$$\mathcal{D}\left(\theta,\mathbf{R}\right) = \sum_{j\in\mathcal{I}}\sum_{i\in\mathcal{I}}\left(\mathbf{S}^{(1,2)}_{ji} - \mathit{sim}(j,i)\right)^2.$$

In this case, the deviation function mathematically expresses the interpretation that $\mathit{sim}(j,i)$ is a good predictor for preferring i if j is also preferred. The key property that determines the analytical solvability of this deviation function is the absence of products of parameters. The non-convex deviation functions in Section 6.1, on the other hand, do contain products of parameters, which are contained in the term $\mathbf{S}_{ui}$. Consequently, they are harder to solve but allow richer parameter interactions.

A typical choice for $\mathit{sim}(j,i)$ is the cosine similarity [10]. The cosine similarity between two items j and i is given by:

$$\mathit{cos}(j,i) = \frac{\sum_{v\in\mathcal{U}}\mathbf{R}_{vj}\mathbf{R}_{vi}}{\sqrt{c(i)\cdot c(j)}}. \tag{50}$$

Another similarity measure is the conditional probability similarity measure [10], which is for two items i and j given by:

$$\mathit{condProb}(j,i) = \sum_{v\in\mathcal{U}}\frac{\mathbf{R}_{vi}\mathbf{R}_{vj}}{c(j)}. \tag{51}$$

Deshpande and Karypis also proposed an adapted version:

$$\mathit{condProb}^*(j,i) = \sum_{v\in\mathcal{U}}\frac{\mathbf{R}_{vi}\mathbf{R}_{vj}}{c(i)\cdot c(j)^{\alpha}\cdot c(v)}, \tag{52}$$

in which $\alpha \in [0,1]$ is a hyperparameter. They introduced the factor $1/c(j)^{\alpha}$ to avoid the recommendation of overly frequent items and the factor $1/c(v)$ to reduce the weight of i and j co-occurring in the preferences of v, if v has more preferences. Other similarity measures were proposed by Aiolli [2]:

$$\mathit{sim}(j,i) = \left(\sum_{v\in\mathcal{U}}\frac{\mathbf{R}_{vi}\mathbf{R}_{vj}}{c(j)^{\alpha}\cdot c(i)^{(1-\alpha)}}\right)^q,$$

with α, q hyperparameters, Gori et al. [18]:

$$\mathit{sim}(j,i) = \frac{\sum_{v\in\mathcal{U}}\mathbf{R}_{vj}\mathbf{R}_{vi}}{\sum_{k\in\mathcal{I}}\sum_{v\in\mathcal{U}}\mathbf{R}_{vj}\mathbf{R}_{vk}},$$

and Wang et al. [68]:

$$\mathit{sim}(j,i) = \log\left(1+\alpha\cdot\frac{\sum_{v\in\mathcal{U}}\mathbf{R}_{vj}\mathbf{R}_{vi}}{c(j)c(i)}\right),$$

with $\alpha \in \mathbb{R}^+_0$ a hyperparameter. Furthermore, Huang et al. show that $\mathit{sim}(j,i)$ can also be chosen from a number of similarity measures that are typically associated with link prediction [22]. Similarly, Bellogin et al. show that typical scoring functions used in information retrieval can also be used for $\mathit{sim}(j,i)$ [4].

It is common practice to introduce sparsity in $\mathbf{S}^{(1,2)}$ by defining

$$\mathit{sim}(j,i) = \mathit{sim}'(j,i)\cdot|\mathit{KNN}(j)\cap\{i\}|, \tag{53}$$

with $\mathit{sim}'(j,i)$ one of the similarity functions defined by Equations 50-52, $\mathit{KNN}(j)$ the set of items l that correspond to the k highest values $\mathit{sim}'(j,l)$, and k a hyperparameter.

Motivated by a qualitative examination of their results, Sigurbjörnsson and Van Zwol [54] proposed additional adaptations:

$$\mathit{sim}(j,i) = s(j)\cdot d(i)\cdot r(j,i)\cdot \mathit{sim}'(j,i)\cdot|\mathit{KNN}(j)\cap\{i\}|,$$

with

$$s(j) = \frac{k_s}{k_s + |k_s - \log c(j)|}, \tag{54}$$

$$d(i) = \frac{k_d}{k_d + |k_d - \log c(i)|}, \tag{55}$$

$$r(j,i) = \frac{k_r}{k_r + (r-1)}, \tag{56}$$

in which i is the r-th most similar item to j and k_s, k_d and k_r are hyperparameters.

Finally, Desphande and Karypis [10] propose to normalize $\mathit{sim}(j,i)$ as

$$\mathit{sim}(j,i) = \frac{\mathit{sim}''(j,i)}{\sum_{l\in\mathcal{I}\setminus\{j\}}\mathit{sim}''(j,l)},$$

with $\mathit{sim}''(j,i)$ defined using Equation 53. Alternatively, Aiolli [2] proposes the normalization

$$\mathit{sim}(j,i) = \frac{\mathit{sim}''(j,i)}{\sum_{l\in\mathcal{I}\setminus\{i\}}\mathit{sim}''(l,i)^{2(1-\beta)}},$$

with β a hyperparameter.

A second set of analytically solvable deviation functions is tailored to the user-based neighborhood factorization model of Equation 13:

$$\mathbf{S} = \mathbf{S}^{(1,1)}\mathbf{R}.$$

In this case, the factor matrix $\mathbf{S}^{(1,1)}$ can be interpreted as a user-similarity matrix. Consequently, these deviation functions compute every parameter in $\mathbf{S}^{(1,1)}$ as

$$\mathbf{S}^{(1,1)}_{uv} = sim(u,v),$$

with $sim(u,v)$ the similarity between users u and v according to some analytically computable similarity function. In the same way as for the item-based case, computing the factor matrix $\mathbf{S}^{(1,1)}_{uv}$ corresponds to minimizing the deviation function

$$\mathcal{D}(\theta,\mathbf{R}) = \sum_{u\in\mathcal{U}}\sum_{v\in\mathcal{U}}\left(\mathbf{S}^{(1,1)}_{uv} - sim(u,v)\right)^2.$$

In this case, the deviation function mathematically expresses the interpretation that users u and v for which $sim(u,v)$ is high, prefer the same items.
Sarwar et al. [48] propose

$$sim(u,v) = |KNN(u) \cap \{v\}|,$$

with $KNN(u)$ the set of users w that have the k highest cosine similarities $\cos(u,w)$ with user u, and k a hyperparameter. In this case, cosine similarity is defined as

$$cos(u,v) = \frac{\sum_{j\in\mathcal{I}}\mathbf{R}_{uj}\mathbf{R}_{vj}}{\sqrt{c(u)\cdot c(v)}}. \tag{57}$$

Alternatively, Aiolli [2] proposes

$$sim(u,v) = \left(\sum_{j\in\mathcal{I}}\frac{\mathbf{R}_{uj}\mathbf{R}_{vj}}{c(u)^{\alpha}\cdot c(v)^{(1-\alpha)}}\right)^q,$$

with α, q hyperparameters, and Wang et al. [68] propose

$$sim(u,v) = \log\left(1+\alpha\cdot\frac{\sum_{j\in\mathcal{U}}\mathbf{R}_{uj}\mathbf{R}_{vj}}{c(u)c(v)}\right),$$

with α a hyperparameter.
The deviation function for the unified neighborhood based factorization model in Equation 14 is given by

$$\mathcal{D}(\theta,\mathbf{R}) = \sum_{u\in\mathcal{U}}\sum_{v\in\mathcal{U}}\left(\mathbf{S}^{(1,1)}_{uv} - sim(u,v)\right)^2 + \sum_{j\in\mathcal{I}}\sum_{i\in\mathcal{I}}\left(\mathbf{S}^{(2,3)}_{ji} - sim(j,i)\right)^2.$$

However, $sim(j,i)$ and $sim(u,v)$ cannot be chosen arbitrarily. Verstrepen and Goethals [67] show that they need to satisfy certain constraints in order to render a well founded unification. Consequently, they propose KUNN, which corresponds to the following similarity definitions that satisfy the necessary constraints:

$$sim(u,v) = \sum_{j\in\mathcal{I}}\frac{\mathbf{R}_{uj}\mathbf{R}_{vj}}{\sqrt{c(u)\cdot c(v)\cdot c(j)}}$$
$$sim(i,j) = \sum_{v\in\mathcal{U}}\frac{\mathbf{R}_{vi}\mathbf{R}_{vj}}{\sqrt{c(i)\cdot c(j)\cdot c(v)}}.$$

5.11.2 *Higher Order Neighborhood-based*

A fourth set of analytically solvable deviation functions is tailored to the higher order itemset-based neighborhood factorization model of Equation 20:

$$\mathbf{S} = \mathbf{X}\mathbf{S}^{(1,2)}.$$

In this case, the deviation function is given by

$$\mathcal{D}(\theta,\mathbf{R}) = \sum_{j\in\mathcal{S}}\sum_{i\in\mathcal{I}}\left(\mathbf{S}^{(1,2)}_{ji} - sim(j,i)\right)^2.$$

with $\mathcal{S} \subseteq 2^{\mathcal{I}}$ the selected itemsets considered in the factorization model.
Deshpande and Karypis [10] propose to define $sim(j,i)$ similarly as for the pairwise interactions (Eq. 53). Alternatively, others [33; 48] proposed

$$sim(j,i) = sim'(j,i)\cdot\max\left(0, c(j\cup\{i\}) - f\right),$$

with f a hyperparameter.
Lin et al. [30] proposed yet another alternative:

$$sim(j,i) = sim'(j,i)\cdot|KNN_c(i)\cap\{j\}| \cdot\max\left(0, condProb(j,i) - c\right),$$

with $KNN_c(i)$ the set of items l that correspond to the k highest values $c(i,l)$, k a hyperparameter, *condProb* the conditional probability according to Equation 51 and c a hyperparameter. Furthermore, they define

$$sim'(j,i) = \frac{\left(\sum_{v\in\mathcal{U}}\mathbf{X}_{vi}\mathbf{X}_{vj}\right)^2}{c(j)}. \tag{58}$$

A fifth and final set of analytically solvable deviation functions is tailored to the higher order userset-based neighborhood factorization model of Equation 21: $\mathbf{S} = \mathbf{S}^{(1,1)}\mathbf{Y}$. In this case, the deviation function is given by $\mathcal{D}(\theta,\mathbf{R}) = \sum_{v\in\mathcal{S}}\sum_{u\in\mathcal{U}}\left(\mathbf{S}^{(1,1)}_{uv} - sim(v,u)\right)^2$, with $\mathcal{S}\subseteq 2^{\mathcal{U}}$ the selected usersets considered in the factorization model. Lin et al. [30] proposed to define

$$sim'(v,u) = \frac{\left(\sum_{j\in\mathcal{I}}\mathbf{Y}_{uj}\mathbf{Y}_{vj}\right)^2}{c(v)}. \tag{59}$$

Alternatively, Symeonidis et al. [60] propose

$$sim(v,u) = \frac{\sum_{j\in\mathcal{I}}\mathbf{Y}_{uj}\mathbf{Y}_{vj}}{c(v)}\cdot|v|\cdot|KNN(u)_{cp}\cap\{v\}|,$$

with $KNN(u)_{cp}$ the set of usersets w that correspond to the k highest values

$$\frac{\sum_{j\in\mathcal{I}}\mathbf{Y}_{uj}\mathbf{Y}_{wj}}{c(w)}.$$

6. MINIMIZATION ALGORITHMS

Efficiently computing the parameters that minimize a deviation function is often non trivial. Furthermore, there is a big difference between minimizing convex and non-convex deviation functions.

6.1 Non-convex Minimization

The two most popular families of minimization algorithms for non-convex deviation functions of collaborative filtering algorithms are *gradient descent* and *alternating least squares*. We discuss both in this section. Furthermore, we also briefly discuss a few other interesting approaches.

6.1.1 *Gradient Descent*

For deviation functions that assume preferences are missing at random, and consequently consider only the known preferences [52], gradient descent (GD) is generally the numerical optimization algorithm of choice. In GD, the parameters θ are randomly initialized. Then, they are iteratively updated in the direction that reduces $\mathcal{D}(\theta, \mathbf{R})$:

$$\theta^{k+1} = \theta^{k} - \eta \nabla \mathcal{D}(\theta, \mathbf{R}),$$

with η a hyperparameter called the learning rate. The update step is larger if the absolute value of the gradient $\nabla \mathcal{D}(\theta, \mathbf{R})$ is larger. A version of GD that converges faster is Stochastic Gradient Descent (SGD). SGD uses the fact that

$$\nabla \mathcal{D}(\theta, \mathbf{R}) = \sum_{t=1}^{T} \nabla \mathcal{D}_t(\mathbf{S}, \mathbf{R}),$$

with T the number of terms in $\mathcal{D}(\mathbf{S}, \mathbf{R})$. Now, instead of computing $\nabla \mathcal{D}(\theta, \mathbf{R})$ in every iteration, only one term t is randomly sampled (with replacement) and the parameters θ are updated as

$$\theta^{k+1} = \theta^{k} - \eta \nabla \mathcal{D}_t(\mathbf{S}, \mathbf{R}).$$

Typically, a convergence criterium of choice is only reached after every term t is sampled multiple times on average.

However, when the deviation function assumes the missing feedback is missing not at random, the summation over the known preferences, $\sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}} \mathbf{R}_{ui}$, is replaced by a summation over all user item pairs, $\sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}}$, and SGD needs to visit approximately 1000 times more terms. This makes the algorithm less attractive for deviation functions that assume the missing feedback is missing not at random.

To mitigate the large number of terms in the gradient, several authors propose to sample the terms in the gradient not uniformly but proportional to their impact on the parameters [75; 44; 76]. These approaches have not only been proven to speed up convergence, but also to improve the quality of the resulting parameters. Weston et al., on the other hand, sample for every known preference i, a number of non preferred items j until they encounter one for which $\mathbf{S}_{uj} + 1 > \mathbf{S}_{ui}$, i.e., it violates the hinge-loss approximation of the ground truth ranking. In this way, they ensure that every update significantly changes the parameters θ [71].

Additionally, due to recent advances in distributed computing infrastructure, parallel and distributed approaches have been proposed to speed up the convergence of SGD-style computations.

One of the classic works is the HogWild algorithm by Recht et al. [43]. In that work, the authors assume that the rating matrix is highly sparse and hence, for any two sampled ratings, their SGD updates will likely be non-conflicting (independent), since any two such updates are unlikely to share either the user or item vectors. As a result, HogWild drops the synchronization requirements, and lets each thread or processor update a random rating value. In the worst case of a conflict, there will be contention in writing out the results. The authors prove convergence of the algorithm under a simple assumption of rating matrix sparsity.

Gemulla et al. [15] present a distributed SGD algorithm (DSGD) in which the main assumption is that some blocks of the rating matrix are mutually independent and hence their variables can be updated in parallel. DSGD uniformly grids the rating matrix R into sub-matrices or blocks, such that they are independent with respect to the rows and columns. This method generates several configurations of the original rating matrix R, which are then fed to the SGD algorithm in sequence. In the actual algorithm, there are two nested for loops. The outer loop selects a configuration C of the independent blocks and sends it to the SGD scheduler. The latter, simply spawns as many parallel threads as the number of independent blocks in each configuration and applies SGD to each of them in parallel. Once the results are back, the next configuration is loaded to each processor after consolidating the results of the last iteration. This process continues until all the configurations are computed.

Zhuang et al. [78] argue that both these methods suffer from two problems. The first is the issue of locking of threads, in which a thread that is executing a slower block needs to finish before the next round of assignments can begin. Second, since the elements of the rating matrix are accessed at random, it may result in high cache-miss and performance degradation. To solve both these problems, the authors propose a shared memory parallel algorithm called Fast Parallel SGD (FPSGD). There are two main features of the algorithm. To overcome the locking issue, the authors suggest continuous execution of SGD blocks by the scheduler. The only two conditions it needs to satisfy are (1) it has to be a free block, and (2) its number of past updates has to be smallest among all the free blocks (random if there is a tie). The second condition is necessary because otherwise all blocks will not get same chance of being updated. For dealing with memory discontinuity, the authors suggest updating each block in a fixed order of users and items. The randomness of the algorithm is introduced in selecting each block for the next update. Since FPSGD always selects the next block in a deterministic fashion, one simple strategy to select the next block in a random fashion is to split the original rating matrix R into many sub blocks (more than the number of available threads). Since many blocks will have the same update number, randomization is needed to select the next block for update. The authors call this method *partial randomization*. The results of experiments on a variety of datasets show that FPSGD achieves lower RMSE in a far smaller number of iterations.

A natural strategy to optimize the update rules in matrix factorization (MF) is to do a block-wise update and let each block be handled by a separate processor/computing unit. This is the strategy proposed by Yin et. al [74]. The authors first show that most of the loss functions used in MF are decomposable in the sense that they are sums over individual loss terms. Hence they can easily be parallelized. Given $A = WH$ as the model, the proposal is to split W and H into $W^{(I)}$ and $H^{(J)}$, such that the total loss can be written as the sum of the losses over indices I and J. Then the task is to develop a block-wise partition rule, in which blocks are updated independently when updating a factor matrix (by fixing the other factor matrix). Each block can be treated as one update unit. There are several ways in which the blocks can be updated. A straightforward way is to update update all blocks of H and then update all blocks of W. This approach can be referred to as concurrent block updates since each update happens concurrently. An alternate strategy is to update some blocks of H and W alternately. This method, called the frequent block-wise update, has the advantage of faster convergence. This happens due to the fact that updated block information are used in each alternating sequence and hence converges faster. The remainder of the paper presents a recipe for implementing these algorithms in MapReduce.

6.1.2 Alternating Least Squares

If the deviation function allows it, alternating least squares (ALS) becomes an interesting alternative to SGD when preferences are assumed to be missing not at random [29; 21]. In this respect, the deviation functions of Equations 25, 30, 31, and 48 are, amongst others, appealing because they can be minimized with a variant of the alternating least squares (ALS) method. Take for example the deviation function from Equation 25:

$$\mathcal{D}\left(\theta, \mathbf{R}\right) = \sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I}} \mathbf{W}_{ui}\left(\mathbf{R}_{ui} - \mathbf{S}_{ui}\right)^2 + \sum_{t=1}^{T}\sum_{f=1}^{F}\lambda_{tf}||\mathbf{S}^{(t,f)}||_F^2,$$

combined with the basic two-factor factorization model from Equation 7:

$$\mathbf{S} = \mathbf{S}^{(1,1)}\mathbf{S}^{(1,2)}.$$

As most deviation functions, this deviation function is non-convex in the parameters θ and has therefore multiple local optima. However, if one temporarily fixes the parameters in $\mathbf{S}^{(1,1)}$, it becomes convex in $\mathbf{S}^{(1,2)}$ and we can analytically find updated values for $\mathbf{S}^{(1,2)}$ that minimize this convex function and are therefore guaranteed to reduce $\mathcal{D}\left(\theta, \mathbf{R}\right)$. Subsequently, one can temporarily fix the parameters in $\mathbf{S}^{(1,2)}$ and in the same way compute updated values for $\mathbf{S}^{(1,1)}$ that are also guaranteed to reduce $\mathcal{D}\left(\theta, \mathbf{R}\right)$. One can keep alternating between fixing $\mathbf{S}^{(1,1)}$ and $\mathbf{S}^{(1,2)}$ until a convergence criterium of choice is met. Hu et al. [21], Pan et al. [37] and Pan and Scholz [36] give detailed descriptions of different ALS variations. The version by Hu et al. contains optimizations for the case in which missing preferences are uniformly weighted. Pan and Scholz [36] describe optimizations that apply to a wider range of weighting schemes. Finally, Pilaszy et al. propose to further speed-up the computation by only approximately solving each convex ALS-step [42].

Additionally, ALS has the advantages that it does not require the tuning of a learning rate, that it can benefit from linear algebra packages such as Intel MKL, and that it needs relatively few iterations to converge. Furthermore, when the basic two-factor factorization of Equation 7 is used, every row of $\mathbf{S}^{(1,1)}$ and every column of $\mathbf{S}^{(1,2)}$ can be updated independently of all other rows or columns, respectively, which makes it fairly easy to massively parallelize the computation of the factor matrices [77].

6.1.3 Bagging

The maximum margin based deviation function in Equation 33 cannot be solved with ALS because it contains the hinge loss. Rennie and Srebro propose a conjugate gradients method for minimizing this function [46]. However, this method suffers from similar problems as SGD, related to the high number of terms in the loss function. Therefore, Pan and Scholz [36] propose a bagging approach. The essence of their bagging approach is that they do not explicitly weight every user-item pair for which $\mathbf{R}_{ui} = 0$, but sample from all these pairs instead. They create multiple samples, and compute multiple different solutions $\widetilde{\mathbf{S}}$ corresponding to their samples. These computations are also performed with the conjugate gradients method. They are, however, much less intensive since they only consider a small sample of the many user-item pairs for which $\mathbf{R}_{ui} = 0$. The different solutions $\widetilde{\mathbf{S}}$ are finally aggregated by simply taking their average.

6.1.4 Coordinate Descent

When an item-based neighborhood model is used in combination with a squared error-based deviation function, the user factors are fixed by definition, and the problem resembles a single ALS step. However, imposing the constraints in Equation 11 complicates the minimization [34]. Therefore, Ning and Karypis adopt cyclic coordinate descent and soft thresholding [14] for SLIM.

6.2 Convex Minimization

Aiolli [3] proposed the convex deviation function in Equation 49 and indicates that it can be solved with any algorithm for convex optimization.

The analytically solvable deviation functions are also convex. Moreover, minimizing them is equivalent to computing all the similarities involved in the model. Most works assume a brute force computation of the similarities. However, Verstrepen [65] recently proposed two methods that are an order of magnitude faster than the brute force computation.

7. RATING BASED METHODS

Interest in collaborative filtering on binary, positive-only data only recently increased. The majority of existing collaborative filtering research assumes rating data. In this case, the feedback of user u about item i, $\mathbf{R}_{ui}$, is an integer between B_l and B_h, with B_l and B_h the most negative and positive feedback, respectively. The most typical example of such data was provided in the context of the Netflix Prize with $B_l = 1$ and $B_h = 5$.

Technically, our case of binary, positive-only data is just a special case of rating data with $B_l = B_h = 1$. However, collaborative filtering methods for rating data are in general built on the implicit assumption that $B_l < B_h$, i.e., that both positive and negative feedback is available. Since this negative feedback is not available in our problem setting, it is not surprising that, in general, methods for rating data generate poor or even nonsensical results [21; 37; 56].

k-NN methods for rating data, for example, often use the Pearson correlation coefficient as a similarity measure. The Pearson correlation coefficient between users u and v is given by

$$pcc(u,v) = \frac{\sum\limits_{\mathbf{R}_{uj},\mathbf{R}_{vj}>0}(\mathbf{R}_{uj} - \overline{\mathbf{R}}_u)(\mathbf{R}_{vj} - \overline{\mathbf{R}}_v)}{\sqrt{\sum\limits_{\mathbf{R}_{uj},\mathbf{R}_{vj}>0}(\mathbf{R}_{uj} - \overline{\mathbf{R}}_u)^2}\sqrt{\sum\limits_{\mathbf{R}_{uj},\mathbf{R}_{vj}>0}(\mathbf{R}_{vj} - \overline{\mathbf{R}}_v)^2}},$$

with $\overline{\mathbf{R}}_u$ and $\overline{\mathbf{R}}_v$ the average rating of u and v respectively. In our setting, with binary, positive-only data however, $\mathbf{R}_{uj}$ and $\mathbf{R}_{vj}$ are by definition always one or zero. Consequently, $\overline{\mathbf{R}}_u$ and $\overline{\mathbf{R}}_v$ are always one. Therefore, the Pearson correlation is always zero or undefined (zero divided by zero), making it a useless similarity measure for binary, positive-only data. Even if we would hack it by omitting the terms for mean centering, $-\overline{\mathbf{R}}_u$ and $-\overline{\mathbf{R}}_v$, it is still useless since it would always be equal to either one or zero.

Furthermore, when computing the score of user u for item i, user(item)-based k-NN methods for rating data typically find the k users (items) that are most similar to u (i) *and that have rated i (have been rated by u)* [11; 23]. On binary, positive-only data, this approach results in the nonsensical result that $\mathbf{S}_{ui} = 1$ for every (u, i)-pair.

The matrix factorization methods for rating data are in general also not applicable to binary, positive-only data. Take for example a basic loss function for matrix factorization on rating data:

$$\min_{\mathbf{S}^{(1)},\mathbf{S}^{(2)}} \sum_{\mathbf{R}_{ui}>0} \left(\mathbf{R}_{ui} - \mathbf{S}^{(1)}_{u\cdot}\mathbf{S}^{(2)}_{\cdot i}\right)^2 + \lambda \left(||\mathbf{S}^{(1)}_{u\cdot}||_F^2 + ||\mathbf{S}^{(2)}_{\cdot i}||_F^2\right),$$

which for binary, positive-only data simplifies to

$$\min_{\mathbf{S}^{(1)},\mathbf{S}^{(2)}} \sum_{\mathbf{R}_{ui}=1} \left(1 - \mathbf{S}^{(1)}_{u\cdot}\mathbf{S}^{(2)}_{\cdot i}\right)^2 + \lambda \left(||\mathbf{S}^{(1)}_{u\cdot}||_F^2 + ||\mathbf{S}^{(2)}_{\cdot i}||_F^2\right).$$

The squared error term of this loss function is minimized when the rows and columns of $\mathbf{S}^{(1)}$ and $\mathbf{S}^{(2)}$, respectively, are all the same unit vector. This is obviously a nonsensical solution.

The matrix factorization method for rating data that uses singular value decomposition to factorize $\mathbf{R}$ also considers the entries where $\mathbf{R}_{ui} = 0$ and does not suffer from the above problem [49; 8]. Although this method does not result in nonsensical results, the performance has been shown inferior to methods specialized for binary, positive-only data [34; 56; 57].

In summary, although we cannot exclude the possibility that there exists a method for rating data that does perform well on binary, positive-only data, in general this is clearly not the case.

8. CONCLUSIONS

We have presented a comprehensive survey of collaborative filtering methods for binary, positive-only data. Its backbone is an innovative unified matrix factorization perspective on collaborative filtering methods, also those that are typically not associated with matrix factorization models such as nearest neighbors methods and association rule mining-based methods. From this perspective, a collaborative filtering algorithm consists of three building blocks: a matrix factorization model, a deviation function and a numerical minimization algorithm. By comparing methods along these three dimensions, we were able to highlight surprising commonalities and key differences.

An interesting direction for future work is to survey certain aspects that were not included in the scope of this survey. Examples are surveying the different strategies to deal with cold-start problems that are applicable to binary, positive-only data; and comparing the applicability of models and deviation functions for recomputation of models in real time upon receiving novel feedback.

9. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. 17(6):734–749, 2005.

[2] F. Aiolli. Efficient top-n recommendation for very large scale binary rated datasets. In *Proc. of the 7th ACM Conf. on Recommender Systems*, pages 273–280. ACM, 2013.

[3] F. Aiolli. Convex auc optimization for top-n recommendation with implicit feedback. In *Proc. of the 8th ACM Conf. on Recommender Systems*, pages 293–296. ACM, 2014.

[4] A. Bellogin, J. Wang, and P. Castells. Text retrieval methods for item ranking in collaborative filtering. In P. Clough, C. Foley, C. Gurrin, G. Jones, W. Kraaij, H. Lee, and V. Mudoch, editors, *Advances in Information Retrieval*, volume 6611, pages 301–306. Springer Berlin Heidelberg, 2011.

[5] C. M. Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *The Journal of machine Learning research*, 3:993–1022, 2003.

[7] E. Christakopoulou and G. Karypis. Hoslim: Higher-order sparse linear method for top-n recommender systems. In *Advances in Knowledge Discovery and Data Mining*, pages 38–49. Springer, 2014.

[8] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proc. of the fourth ACM Conf. on Recommender Systems*, pages 39–46. ACM, 2010.

[9] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[10] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[11] C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*, pages 107–144. Springer, 2011.

[12] C. Dhanjal, R. Gaudel, and S. Clémençon. Collaborative filtering with localised ranking. In *Proc. of the 29th AAAI Conf. on Artificial Intelligence*, 2015.

[13] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[14] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[15] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proc. of the 17th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 69–77, 2011.

[16] P. Gopalan, J. Hofman, and D. Blei. Scalable recommendation with hierarchical poisson factorization. In *Proc. of the 31st Conf. Annual Conf. on Uncertainty in Artificial Intelligence (UAI-15). AUAI Press*, 2015.

[17] P. Gopalan, F. J. Ruiz, R. Ranganath, and D. M. Blei. Bayesian nonparametric poisson factorization for recommendation systems. *Artificial Intelligence and Statistics (AISTATS)*, 33:275–283, 2014.

[18] M. Gori, A. Pucci, V. Roma, and I. Siena. Itemrank: A random-walk based scoring algorithm for recommender engines. In *Proc. of the 20th Int. Joint Conf. on Artificial Intelligence*, volume 7, pages 2766–2771, 2007.

[19] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115, 2004.

[20] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence*, pages 688–693, 1999.

[21] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proc. of the Eighth IEEE Int. Conf. on Data Mining*, pages 263–272. IEEE, 2008.

[22] Z. Huang, X. Li, and H. Chen. Link prediction approach to collaborative filtering. In *Proc. of the 5th ACM/IEEE-CS joint Conf. on Digital libraries*, pages 141–142. ACM, 2005.

[23] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010.

[24] C. Johnson. Logistic matrix factorization for implicit feedback data. In *Workshop on Distributed Machine Learning and Matrix Computations at the Twenty-eighth Annual Conf. on Neural Information Processing Systems (NIPS)*, 2014.

[25] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[26] S. Kabbur and G. Karypis. Nlmf: Nonlinear matrix factorization methods for top-n recommender systems. In *Workshop Proc. of the IEEE Int. Conf. on Data Mining*, pages 167–174. IEEE, 2014.

[27] S. Kabbur, X. Ning, and G. Karypis. Fism: factored item similarity models for top-n recommender systems. In *Proc. of the 19th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*, pages 659–667. ACM, 2013.

[28] N. Koenigstein, N. Nice, U. Paquet, and N. Schleyen. The xbox recommender system. In *Proc. of the sixth ACM Conf. on Recommender Systems*, pages 281–284. ACM, 2012.

[29] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[30] W. Lin, S. A. Alvarez, and C. Ruiz. Efficient adaptive-support association rule mining for recommender systems. *Data mining and knowledge discovery*, 6(1):83–105, 2002.

[31] G. V. Menezes, J. M. Almeida, F. Belém, M. A. Gonçalves, A. Lacerda, E. S. De Moura, G. L. Pappa, A. Veloso, and N. Ziviani. Demand-driven tag recommendation. In *Machine Learning and Knowledge Discovery in Databases*, pages 402–417. Springer, 2010.

[32] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[33] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Effective personalization based on association rule discovery from web usage data. In *Proc. of the 3rd Int. workshop on Web information and data management*, pages 9–15. ACM, 2001.

[34] X. Ning and G. Karypis. Slim: Sparse linear methods for top-n recommender systems. In *Proc. of the 11th IEEE Int. Conf. on Data Mining*, pages 497–506. IEEE, 2011.

[35] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[36] R. Pan and M. Scholz. Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering. In *Proc. of the 15th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*, pages 667–676. ACM, 2009.

[37] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Proc. of the Eighth IEEE Int. Conf. on Data Mining*, pages 502–511. IEEE, 2008.

[38] W. Pan and L. Chen. Cofiset: Collaborative filtering via learning pairwise preferences over item-sets. In *Proc. of the 13th SIAM Int. Conf. on Data Mining*, pages 180–188, 2013.

[39] W. Pan and L. Chen. Gbpr: Group preference based bayesian personalized ranking for one-class collaborative filtering. In *Proc. of the Twenty-Third Int. joint Conf. on Artificial Intelligence*, pages 2691–2697. AAAI Press, 2013.

[40] U. Paquet and N. Koenigstein. One-class collaborative filtering with random graphs. In *Proc. of the 22nd Int. Conf. on WWW*, pages 999–1008, 2013.

[41] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proc. of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[42] I. Pilászy, D. Zibriczky, and D. Tikk. Fast als-based matrix factorization for explicit and implicit feedback datasets. In *Proc. of the fourth ACM Conf. on Recommender Systems*, pages 71–78. ACM, 2010.

[43] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems 24*, pages 693–701. 2011.

[44] S. Rendle and C. Freudenthaler. Improving pairwise learning for item recommendation from implicit feedback. In *Proc. of the 7th ACM Int. Conf. on Web search and data mining*, pages 273–282. ACM, 2014.

[45] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proc. of the Twenty-Fifth Conf. on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[46] J. D. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proc. of the 22nd Int. Conf. on Machine learning*, pages 713–719. ACM, 2005.

[47] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. *Recommender Systems Handbook*. Springer, Boston, MA, 2011.

[48] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *Proc. of the 2nd ACM Conf. on Electronic commerce*, pages 158–167. ACM, 2000.

[49] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[50] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proc. of the 10th Int. Conf. on WWW*, pages 285–295. ACM, 2001.

[51] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, A. Hanjalic, and N. Oliver. Tfmap: Optimizing map for top-n context-aware recommendation. In *Proc. of the 35th Int. ACM SIGIR Conf. on Research and development in information retrieval*, pages 155–164. ACM, 2012.

[52] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proc. of the sixth ACM Conf. on Recommender Systems*, pages 139–146. ACM, 2012.

[53] Y. Shi, M. Larson, and A. Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):3, 2014.

[54] B. Sigurbjörnsson and R. Van Zwol. Flickr tag recommendation based on collective knowledge. In *The 17th Int. Conf. on WWW*, pages 327–336. ACM, 2008.

[55] V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic. One-class matrix completion with low-density factorizations. In *Proc. of the 10th IEEE Int. Conf. on Data Mining*, pages 1055–1060. IEEE, 2010.

[56] H. Steck. Training and testing of recommender systems on data missing not at random. In *Proc. of the 16th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*, pages 713–722. ACM, 2010.

[57] H. Steck. Item popularity and recommendation accuracy. In *Proc. of the fifth ACM Conf. on Recommender Systems*, pages 125–132. ACM, 2011.

[58] H. Steck. Gaussian ranking by matrix factorization. In *Proc. of the 9th ACM Conf. on Recommender Systems*, pages 115–122. ACM, 2015.

[59] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.

[60] P. Symeonidis, A. Nanopoulos, A. N. Papadopoulos, and Y. Manolopoulos. Nearest-biclusters collaborative filtering based on constant and coherent values. *Inf. Retr.*, 11(1):51–75, 2008.

[61] G. Takács and D. Tikk. Alternating least squares for personalized ranking. In *Proc. of the sixth ACM Conf. on Recommender Systems*, pages 83–90. ACM, 2012.

[62] A. Töscher and M. Jahrer. Collaborative filtering ensemble for ranking. *Journal of Machine Learning Research W&CP: Proc. of KDD Cup 2011*, 18:61–74, 2012.

[63] L. H. Ungar and D. P. Foster. Clustering methods for collaborative filtering. In *AAAI workshop on recommendation systems*, volume 1, pages 114–129, 1998.

[64] M. van Leeuwen and D. Puspitaningrum. Improving tag recommendation using few associations. In *Advances in Intelligent Data Analysis XI*, pages 184–194. Springer, 2012.

[65] K. Verstrepen. *Collaborative Filtering with Binary, Positive-Only Data.* PhD thesis, University of Antwerp, 2015.

[66] K. Verstrepen and B. Goethals. Unifying nearest neighbors collaborative filtering. In *Proc. of the 8th ACM Conf. on Recommender Systems*, pages 177–184. ACM, 2014.

[67] K. Verstrepen and B. Goethals. Top-n recommendation for shared accounts. In *Proc. of the 9th ACM Conf. on Recommender Systems*, pages 59–66. ACM, 2015.

[68] J. Wang, A. P. De Vries, and M. J. Reinders. A user-item relevance model for log-based collaborative filtering. In *Advances in Information Retrieval*, pages 37–48. Springer, 2006.

[69] J. Weston, S. Bengio, and N. Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *Proc. of the 22th Interantional Joint Conf. on Artifical Intelligence*, volume 11, pages 2764–2770, 2011.

[70] J. Weston, R. J. Weiss, and H. Yee. Nonlinear latent factorization by embedding multiple user interests. In *Proc. of the 7th ACM Conf. on Recommender Systems*, pages 65–68. ACM, 2013.

[71] J. Weston, H. Yee, and R. J. Weiss. Learning to rank recommendations with the k-order statistic loss. In *Proc. of the 7th ACM Conf. on Recommender Systems*, pages 245–248. ACM, 2013.

[72] X. Yang, Y. Guo, Y. Liu, and H. Steck. A survey of collaborative filtering based social recommender systems. *Computer Communications*, 41:1–10, 2014.

[73] Y. Yao, H. Tong, G. Yan, F. Xu, X. Zhang, B. K. Szymanski, and J. Lu. Dual-regularized one-class collaborative filtering. In *Proc. of the 23rd ACM Int. Conf. on Information and Knowledge Management*, pages 759–768. ACM, 2014.

[74] J. Yin, L. Gao, and Z. M. Zhang. *Machine Learning and Knowledge Discovery in Databases: European Conf., ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proc., Part III*, chapter Scalable Nonnegative Matrix Factorization with Block-wise Updates, pages 337–352. 2014.

[75] W. Zhang, T. Chen, J. Wang, and Y. Yu. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proc. of the 36th Int. ACM SIGIR Conf. on Research and development in information retrieval*, pages 785–788. ACM, 2013.

[76] H. Zhong, W. Pan, C. Xu, Z. Yin, and Z. Ming. Adaptive pairwise preference learning for collaborative recommendation with implicit feedbacks. In *Proc. of the 23rd ACM Int. Conf. on Conf. on Information and Knowledge Management*, pages 1999–2002. ACM, 2014.

[77] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Algorithmic Aspects in Information and Management*, pages 337–348. Springer, 2008.

[78] Y. Zhuang, W.-S. Chin, Y.-C. Juan, and C.-J. Lin. A fast parallel sgd for matrix factorization in shared memory systems. In *Proc. of the 7th ACM Conf. on Recommender Systems*, pages 249–256, 2013.