Expediting Exploration by Attribute-to-Feature Mapping for Cold-Start Recommendations

Deborah Cohen*
Google Research
Tel-Aviv, Israel
debbycohen@gmail.com

Michal Aharon, Yair Koren, and Oren Somekh
Yahoo Research, Haifa, Israel
(michala,yairkoren,orens)
@yahoo-inc.com

Raz Nissim*
General Motors
Advanced Technical Center
Herzliya, Israel
raz.nissim@gm.com

ABSTRACT

The item cold-start problem is inherent to collaborative filtering (CF) recommenders where items and users are represented by vectors in a latent space. It emerges since CF recommenders rely solely on historical user interactions to characterize their item inventory. As a result, an effective serving of new and trendy items to users may be delayed until enough user feedback is received, thus, reducing both users' and content suppliers' satisfaction. To mitigate this problem, many commercial recommenders apply random exploration and devote a small portion of their traffic to explore new items and gather interactions from random users. Alternatively, content or context information is combined into the CF recommender, resulting in a hybrid system. Another hybrid approach is to learn a mapping between the item attribute space and the CF latent feature space, and use it to characterize the new items providing initial estimates for their latent vectors.

In this paper, we adopt the attribute-to-feature mapping approach to expedite random exploration of new items and present LearnAROMA - an advanced algorithm for learning the mapping, previously proposed in the context of classification. In particular, LearnAROMA learns a Gaussian distribution over the mapping matrix. Numerical evaluation demonstrates that this learning technique achieves more accurate initial estimates than logistic regression methods. We then consider a random exploration setting, in which new items are further explored as user interactions arrive. To leverage the initial latent vector estimates with the incoming interactions, we propose DynamicBPR - an algorithm for updating the new item latent vectors without retraining the CF model. Numerical evaluation reveals that DynamicBPR achieves similar accuracy as a CF model trained on all the ratings, using 71% less exploring users than conventional random exploration.

KEYWORDS

Recommendation systems, collaborative-filtering, random exploration, item cold-start problem

*This work was done while Deborah and Raz were with Yahoo Research, Haifa, Israel.

RecSys'17, August 27–31, 2017, Como, Italy.
 ISBN 978-1-4503-4652-8/17/08…$15.00.
DOI: http://dx.doi.org/10.1145/3109859.3109880

1 INTRODUCTION

In recent years, recommendation technologies have become an essential tool for matching relevant content and services to users. Whether these are news items, movies, mobile apps, or even ads, recommender systems usually use historical user interactions and stated preferences, and content information to characterize their items. Techniques that rely on content information are known as *content-based* [18], while techniques that rely solely on historical users interactions, such as clicks, skips and ratings, are known as *collaborative filtering* (CF) [14]. Notably, CF techniques require no domain knowledge or content analysis and excel at exploiting popularity trends, which drive much of the observed user interactions and typically would be completely missed by content based approaches. In addition, relying directly on user behavior allows uncovering complex and unexpected patterns that would be difficult or impossible to profile using known data attributes. Therefore, CF attracted much attention in recent years, resulting in significant progress and adoption by many commercial systems.

Since CF recommenders rely only on historical user interactions to characterize their entities, the lack of such data for new entities imposes an inherent problem known as the *cold-start problem*. Hence, without sufficient historical user data, CF-based recommenders cannot reliably model new entities. Despite the fact that users and items have similar representations in the latent space, the users and items cold-start problems are essentially different. The main difference comes from the ability to interview new users when joining a service in order to bootstrap their modeling. Besides, in most settings the number of users is much larger than the number of items. Hence, a typical item usually gets more ratings than an individual user provides.

To mitigate the CF item cold-start problem, a few known approaches are usually applied. The most common method, used by many commercial web-scale recommenders, is random exploration, where a small random portion of the user traffic is devoted to explore new items. Then, the resulting user feedback is used to model the new items. This approach is simple but inefficient, since the matching between users and items is almost arbitrary and recent work shows that carefully selecting the exploring users is beneficial [1, 6]. Another common approach to combat the item cold-start issue is to utilize item attributes in addition to the users feedback resulting in a *hybrid* recommender. Numerous ways to integrate the item attributes into the CF model have been suggested over the years. The basic principle driving this approach is that new items share attributes with mature items. Therefore, the recommender can characterize the latter with latent vector representations of their attributes and provide an initial representation for the new

items. Another approach which exploits item attributes as well, but in an implicit way, is presented in [8]. In particular, the authors train a CF model and then learn a linear mapping between the item attributes space and the CF latent space. Once the mapping is set, it is used to provide an initial estimate for the new items latent vectors.

In this paper, we address the CF item cold-start problem and adopt a hybrid approach. The presentation is formulated in terms of item cold-starts since this problem is typically more challenging. However, the derivations and algorithms can be directly applied to user cold-starts as well. Inspired by the approach of [8], we consider item attributes to CF latent space mappings. In particular, in order to learn the linear mapping between attributes and latent features, we present LearnAROMA, a variant of the AROMA (Adaptive Regularization Of MAtrix models) algorithm, presented in [4, 5] in the context of linear classification. The key principle of AROMA is that instead of maintaining a single weight matrix, it considers a Gaussian distribution over possible models. Moreover, while [8] considers an initial estimate to the new item latent vectors only, we also consider a random exploration setting where users interactions with the new items are arriving. For this setting, we present a simple algorithm, DynamicBPR, that updates the new items latent vectors with the incoming interactions, using a regularized maximum likelihood convex objective. Our approach does not require to retrain the CF model, which would be inefficient in large scale systems. Finally, we test our LearnAROMA based initial estimate and update algorithm DynamicBPR using the MovieLens2K dataset, which includes genre, and crew information in addition to users ratings, and demonstrate their benefits over several baselines. The main contributions of this work are:

- The LearnAROMA algorithm for learning items attributes to latent space mapping, that provides an initial estimate of new items latent vectors.
- The DynamicBPR algorithm for updating new items latent vectors during random exploration, leveraging their initial estimate with incoming users' interactions, without retraining the whole CF model.
- Numerical evaluation that demonstrates the superiority of the LearnAROMA algorithm over the LearnMap algorithm of [8]. In addition, the DynamicBPR algorithm is shown to achieve similar accuracies to those achieved by retraining the whole model, requiring far less interactions compared to random exploration.

The rest of the paper is organized as follows. In Section 2, we review related work. Section 3 formulates the item cold-start problem and sets our goals. In Section 4, we describe LearnMap, an item attributes to latent space mapping approach, to mitigate the CF item cold-start problem. An alternative algorithm to learn this mapping, LearnAROMA, is elaborated in Section 5. Section 6 formulates the exploration setting and presents the DynamicBPR algorithm that dynamically updates the latent vectors. Numerical evaluation results are presented in Section 7. We conclude and consider future work in Section 8.

2 RELATED WORK

Several approaches have recently been proposed, that aim at mitigating the inherent item cold-start problem in CF recommenders. These can be divided into two main categories; pure CF systems that gather exploratory feedback from users on new items, and hybrid approaches that combine content-based and CF recommendation techniques.

2.1 Pure CF Techniques

A naive common approach for this first category is to use an exploration bucket, where randomly selected users are exposed to the new item, generating feedback for the latter. This simple and inefficient approach can be improved by selecting the most useful users, who are more likely to be interested in the new items [6], or who will provide the best characterization for them [1, 15]. Unfortunately, these approaches require user cooperation.

Other CF techniques address the wide sense cold-start users, that is infrequent users with few ratings [13, 22]. In this setting, the users latent vectors express similarities depending on the items rated by the user. However, these can obviously not be applied to the narrow sense cold-starts problem, where no ratings are available for new entities (see [21] for online update of items with few ratings).

Another approach that combines exploration-exploitation simultaneously, is based on *multi-armed bandit* (MAB) where presenting an item to a user in a specific context is abstracted as pulling a MAB arm with some uncertain reward (cf., [3] and references therein). However, these methods become very complex when the item inventory is large (e.g., typical ad inventory can reach 200K items).

2.2 Hybrid Techniques

Hybrid techniques alleviate the need for users' help by exploiting content information. Several strategies can be adopted for creating hybrid techniques: working directly in the item and user attribute space [10, 11, 16, 17]; incorporating item attributes into the latent space [19]; mapping the attribute space to the CF latent space in order to estimate new items' latent vectors [2, 8].

Avoiding asking real users for their help, [17] uses artificial users or bots implementing various heuristics, in part based on content information, to generate synthetic data. A stochastic hybrid approach, where the joint distribution of users vote vector is modeled using a *Boltzmann machine*, is presented in [10, 11].

In [2], a similar framework as in [22] is considered, where the mean prior of the item and user latent vectors are expressed as linear functions of their corresponding known attributes. Here, given the ratings and attributes, the goal is to find the maximum likelihood estimate of the parameters that define the mapping between the attribute and latent domains. In addition, the authors consider an incremental learning setting where for each batch of observations, the current posterior becomes prior and is combined with the data likelihood to provide an updated posterior.

A general approach referred to as *factorization machines* is presented in [19]. Special cases of the factorization machines include but are not limited to the traditional *matrix factorization* (MF), SVD++ (see [13]), nearest neighbor models and attribute-aware models. In the latter case, each attribute is associated with a latent vector which are directly included in the feature vectors.

A hybrid MF-CF and content-based technique is adopted in [8]. The goal is to learn the mapping function between the entities (items) attribute space and their latent space. The factorization

model is first trained on the known ratings. Then, two learning approaches are suggested; the first is a *K-Nearest Neighbor* (KNN) mapping where the new latent vectors are weighted averages of the features of the k closest neighbors in the attribute space. The second approach assumes that the latent vectors are linear functions of the attributes. Once the mapping functions are found, the latent vector of a new entity with known attributes can be computed and new item-user scores can be derived by taking the inner product between corresponding user-item latent vectors (see also [23] where the mapping approach is used sub-optimally as a baseline).

3 PROBLEM FORMULATION

In this section, we present the recommendation model and formulate the item cold-start issue. We then describe our generic framework and MF model training.

3.1 Model Description

Consider a system with N users and M items, including M_1 mature items and M_2 new items (cold-starts), so that $M = M_1 + M_2$. Each item is associated with a known $L \times 1$ binary attribute vector $\mathbf{a}_j$ that contains the item content information (e.g., for movies: genres, actors and directors). For example, consider a recommender system for movies including the movies *The Butterfly Effect* and *Toy Story* and the following $L = 4$ genres: *Animation, Comedy, Drama, Fantasy*. If we assign consecutive indices in alphabetical order to both the movies and genres, the resulting attributes vectors are $\mathbf{a}_1 = [0\ 0\ 1\ 0]^T$ and $\mathbf{a}_2 = [1\ 1\ 0\ 1]^T$. Attributes representation is not restricted to a binary set and the mapping derived here for binary attributes, for the sake of simplicity, can be extended to other attributes forms.

Let $\mathbf{S}$ be the $N \times M$ binary feedback matrix, with $\mathbf{S}_{ij} = 1$ if user i has provided positive feedback on item j; otherwise $\mathbf{S}_{ij} = 0$. Positive feedback refers to any user-item interaction, e.g. rating or click, while negative feedback indicates a lack of interaction. Denote the set of items for which user i has provided positive feedback by $I_i^+ = \{j|\mathbf{S}_{ij} = 1\}$ and the set of items for which user i has not provided feedback by $I_i^- = \{j|\mathbf{S}_{ij} = 0\}$. The feedback for mature items, namely $\mathbf{S}_{ij}$, for $1 \le j \le M_1$ is assumed to be known, whereas it is dynamically discovered for new (or cold-start) items, $M_1 < j \le M$, one user-item interaction at a time.

In terms of items, our dataset is divided between mature and new items. For new items, we distinguish between users that provide feedback allowing for dynamic training and test users. We thus divide the dataset into 3 parts, as shown in Fig. 1. The first group $\mathcal{A}_1$ constitutes the model training group and is composed of the N users and the M_1 mature items. The second group $\mathcal{A}_2$ is composed of N_1 users and the M_2 new items for dynamic training. Lastly, the third group $\mathcal{A}_3$ represents the test group and includes the remaining interactions (or feedback) between the $N_2 = N - N_1$ users and M_2 new items.

Our goal is twofold. First, we provide an initial estimate for the latent vectors of new items, before any feedback for them is acquired. Second, we dynamically update these latent vectors with respect to sequentially incoming feedback without retraining the model.

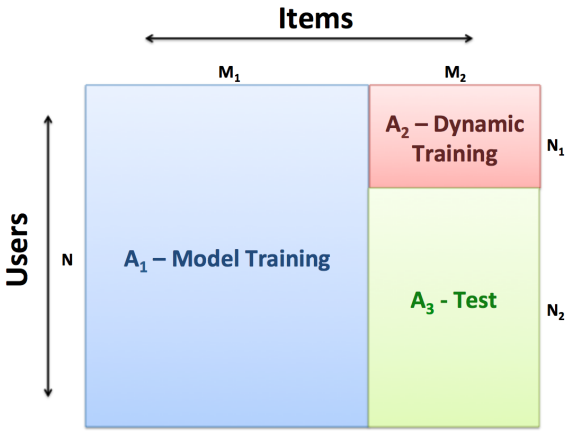

Figure 1: Dataset division.

3.2 High-Level Framework

Our approach includes three stages. The first is a traditional MF that trains a static and stable model composed of users and mature items, as described earlier. This stage results in latent vectors $\mathbf{u}_i$, $1 \le i \le N$ and $\mathbf{v}_j$, $1 \le j \le M_1$, for users and items, respectively. The second stage derives a mapping between item content-based attributes $\mathbf{a}_j$ and latent vectors $\mathbf{v}_j$ computed in the previous stage, yielding a mapping matrix $\mathbf{W}$. Using the attribute-to-feature mapping, we can then compute initial latent vectors $\hat{\mathbf{v}}_j^0 = \mathbf{W}\mathbf{a}_j$ for new items, or cold-starts, namely $M_1 < j \le M$. The last stage dynamically updates the new mapped latent vectors $\hat{\mathbf{v}}_j$ from new feedback on cold-starts items. This dynamic training of cold-start items does not require retraining the mature model, but only expands it to include new items.

In the next section, we describe the MF approach we adopt. In Sections 4 and 5, we present two attribute-to-feature mapping methods. Last, in Section 6, we show how the latent vectors can be dynamically updated.

3.3 BPR-MF Model Training

The principle of MF is to represent both items and users in a common latent space, which aims at capturing patterns of users feedback on items. More specifically, the latent vectors $\mathbf{u}_i$, $1 \le i \le N$ and $\mathbf{v}_j$, $1 \le j \le M_1$, for users and items, respectively, are inferred from the feedback such that the corresponding score of user i to item j is estimated by their inner product

$$\hat{y}_{ij} = \langle \mathbf{u}_i, \mathbf{v}_j \rangle + b_i + b_j = \sum_{k=1}^{K} u_{ik} v_{jk} + b_i + b_j. \quad (1)$$

Here, K is the latent space dimension, u_{ik} (v_{jk}) denotes the kth latent factor (or latent feature) of vector $\mathbf{u}_i$ ($\mathbf{v}_j$), and b_i and b_j are the biases of user i and item j, respectively. These users-items scores are used to rank the items for a specific user.

Specifically, we consider the *Bayesian personalized ranking* (BPR) [20] framework, used for model training with implicit feedback. The key idea is to consider entity pairs instead of single entities in the loss function, allowing the interpretation of positive-only data as partial ranking data and canceling the impact of entity bias. Here,

we consider pairs of items per user, where the first item received positive feedback from the user and the second one did not. The BPR-MF optimization criterion, expressed as follows

$$\max_{\mathbf{u}_i,\mathbf{v}_p,\mathbf{v}_q,b_p,b_q} \sum_{i,p,q\in\mathcal{A}_1} \ln \frac{1}{1+e^{-\hat{x}_{ipq}}} - \lambda\left(||\mathbf{u}_i||^2 + ||\mathbf{v}_p||^2 + ||\mathbf{v}_q||^2\right) - \lambda_b\left(b_p^2 + b_q^2\right) \quad (2)$$

is a non-convex optimization problem. Here, $\lambda, \lambda_b > 0$ are regularization parameters, $\mathcal{A}_1 = \left\{(i,p,q)|p \in I_i^+ \wedge q \in I_i^-\right\}$ and

$$\hat{x}_{ipq} = \hat{y}_{ip} - \hat{y}_{iq}. \quad (3)$$

Note that in this case, the user bias b_i in (1) cancels out; thus, we do not consider user biases.

As the optimization criterion (2) is differentiable, an obvious choice for maximization is gradient descent. Since full gradient descent suffers from slow convergence, the stochastic approach is adopted in [20]. It has been empirically shown [20] that the order in which the training pairs are traversed is crucial. In particular, an item-wise or user-wise traverse approach leads to poor convergence whereas the proposed bootstrap sampling method, that is drawing user-items triples uniformly at random, converges much faster. In order to learn the BPR-MF model, we thus use the LearnBPR algorithm from [20], summarized in Algorithm 1.

Algorithm 1 LearnBPR

Input:
S($\mathcal{A}_1$) - feedback of N users on M_1 mature items
$\mathbf{v}_j$, b_j, $1 \le j \le M_1$ - mature items latent vectors and biases
α, α_b - update step sizes
λ, λ_b - regularization parameters
Output:
$\mathbf{u}_i$, $1 \le i \le N$ - users latent vectors

repeat
  draw (i,p,q) from $\mathcal{A}_1$
  $\hat{x}_{ipq} \leftarrow \hat{y}_{ip} - \hat{y}_{iq}$
  for $1 \le k \le K$ **do**
    $u_{ik} \leftarrow u_{ik} + \alpha\left(\frac{e^{-\hat{x}_{ipq}}}{1+e^{-\hat{x}_{ipq}}}(v_{pk} - v_{qk}) - \lambda u_{ik}\right)$
    $v_{pk} \leftarrow v_{pk} + \alpha\left(\frac{e^{-\hat{x}_{ipq}}}{1+e^{-\hat{x}_{ipq}}}u_{ik} - \lambda v_{pk}\right)$
    $v_{qk} \leftarrow v_{qk} + \alpha\left(-\frac{e^{-\hat{x}_{ipq}}}{1+e^{-\hat{x}_{ipq}}}u_{ik} - \lambda v_{qk}\right)$
    $b_p \leftarrow b_p + \alpha_b\left(\frac{e^{-\hat{x}_{ipq}}}{1+e^{-\hat{x}_{ipq}}} - \lambda_b b_p\right)$
    $b_q \leftarrow b_q + \alpha_b\left(-\frac{e^{-\hat{x}_{ipq}}}{1+e^{-\hat{x}_{ipq}}} - \lambda_b b_q\right)$
  end for
until convergence

4 ATTRIBUTE-TO-FEATURE MAPPING

The first stage of the training, described in Section 3.3, is a traditional BPR-MF whose output are latent vectors $\mathbf{u}_i$ for each user $1 \le i \le N$ and $\mathbf{v}_j$ for each mature item $1 \le j \le M_1$. The next training stage maps the item attribute vectors $\mathbf{a}_j$ to the item latent space, such that

$$\mathbf{v}_j = \mathbf{h}(\mathbf{a}_j), \quad (4)$$

for some function $\mathbf{h} : \mathbb{Z}_2^L \to \mathbb{R}^K$, where $\mathbb{Z}_2 = \{0,1\}$. We focus on a linear mapping, where the mapping function is represented by a $K \times L$ matrix. In this stage, we learn the mapping matrix $\mathbf{W}$.

In this case, the inner product between latent vectors becomes

$$\hat{y}_{ij} = \langle \mathbf{u}_i, \mathbf{v}_j \rangle = \mathbf{u}_i^T \mathbf{W} \mathbf{a}_j. \quad (5)$$

The convex optimization criterion with respect to $\mathbf{W}$ is then expressed as follows

$$\max_{\mathbf{W}} \sum_{i,p,q\in A} \ln \frac{1}{1+e^{-\hat{x}_{ipq}}} - \lambda||\mathbf{W}||_F, \quad (6)$$

where $||\cdot||_F$ denotes the *Frobenius norm* and $\hat{x}_{ipq} = \hat{y}_{ip} - \hat{y}_{iq}$, with $\hat{y}_{ip}$ and $\hat{y}_{iq}$ defined in (5).

In order to learn the mapping matrix $\mathbf{W}$, the LearnMap algorithm from [8], summarized in Algorithm 2 for completeness, uses a gradient descent approach. In the algorithm description, $\mathbf{w}^k$ denotes the kth row of $\mathbf{W}$.

Algorithm 2 LearnMap

Input:
S($\mathcal{A}_1$) - feedback of N users on M_1 mature items
$\mathbf{u}_i$, $1 \le i \le N$ - users latent vectors
$\mathbf{a}_j$, $1 \le j \le M_1$ - mature items attribute vectors
α - update step size
λ - regularization parameter
Output:
$\mathbf{W}$ - $K \times L$ attribute-to-feature mapping matrix

repeat
  draw (i,p,q) from $\mathcal{A}_1$
  $\hat{x}_{ipq} \leftarrow \hat{y}_{ip} - \hat{y}_{iq}$
  for $1 \le k \le K$ **do**
    $\mathbf{w}^k \leftarrow \mathbf{w}^k + \alpha\left(\frac{e^{-\hat{x}_{ipq}}}{1+e^{-\hat{x}_{ipq}}}u_{ik}(\mathbf{a}_p - \mathbf{a}_1) - \lambda\mathbf{w}^k\right)$
  end for
until convergence

Once $\mathbf{W}$ is learnt, the resulting item latent vectors are then given by

$$\hat{\mathbf{v}}_j^0 = \mathbf{W}\mathbf{a}_j. \quad (7)$$

The mapped item latent vectors $\hat{\mathbf{v}}_j^0$, for $M_1 < j \le M$, are our initial estimate for new, or cold-start, items.

It is noted that the linear approach can be extended to non-linear mapping to account for interactions between attributes. However, we found that non-linear mappings (e.g., quadratic polynomial mapping) yield marginal improvement.

5 AROMA-BASED MAPPING

In this section, we present an alternative mapping learning algorithm based on the *adaptive regularization of matrix models* (AROMA) algorithm, presented in [4] in the context of linear classification. The matrix approach AROMA extends its vector version *adaptive regularization of weights* (AROW) [5] to learn a weight matrix $\mathbf{W}$ and its corresponding confidence coefficient matrix Σ. The key idea of AROW is that instead of maintaining a single weight vector $\mathbf{w}$, it considers distribution over possible models. In particular, AROW maintains a Gaussian distribution over vectors denoted by $\mathcal{N}(\mathbf{w}, \Sigma)$.

5.1 Preliminaries

The goal of AROMA is to learn a linear similarity measure between pairs of objects - here, items and users - in the form $S_W(\mathbf{u}, \mathbf{a}) = \mathbf{u}_i^T \mathbf{W} \mathbf{a}_j$. A weak supervision setup is adopted where training is based on relative similarity. In particular, at the tth round, a triplet $\mathbf{u}_t = \mathbf{u}_i$, for some $1 \leq i \leq N$, $\mathbf{a}_t^+ \in I_i^+, \mathbf{a}_t^- \in I_i^-$ is sampled. The objective is to order the similarity measures of the objects $\mathbf{a}_t^+, \mathbf{a}_t^-$ with a safety margin

$$S_W(\mathbf{u}_t, \mathbf{a}_t^+) \geq S_W(\mathbf{u}_t, \mathbf{a}_t^-) + 1. \tag{8}$$

To learn a scoring function that obeys (8), the authors use the following *squared-hinge loss*, or L2 loss,

$$\ell_W^2(\mathbf{u}_t, \mathbf{a}_t^+, \mathbf{a}_t^-) = \left(\max\{0, 1 - \mathbf{u}_t^T \mathbf{W}(\mathbf{a}_t^+ - \mathbf{a}_t^-)\}\right)^2, \tag{9}$$

which is a common alternative to the ℓ_1 loss in binary classification. Two special covariance matrix cases are considered. The first assumes a diagonal covariance matrix Σ which leads to a diagonal mapping matrix $\mathbf{W}$. We thus focus on the second case, referred to as factored covariance [4]. This approach models the distribution of mapping matrices based on factorizing the covariance matrix to capture separately correlations in the "input" (right side of the mapping matrix, i.e., attributes in our setting) and in the "output" (left side, i.e., latent vectors) [4]. To that end, the authors in [4] use the definition of a *matrix variate normal distribution* [12]. In Definition 5.1, $\otimes$ denotes the *Kronecker product*.

Definition 5.1. A random matrix $\mathbf{X} \in \mathbb{R}^{K \times L}$ is said to have a matrix variate normal distribution with mean matrix $\mathbf{W} \in \mathbb{R}^{K \times L}$ and covariance matrix $\Omega \otimes \Lambda$, where $\Omega \in \mathbb{R}^{K \times K}$ and $\Lambda \in \mathbb{R}^{L \times L}$ are both symmetric and positive semidefinite (PSD), if $\text{vec}(\mathbf{X}) \sim \mathcal{N}(\text{vec}(\mathbf{W}), \Omega \otimes \Lambda)$. Matrix variate normal distributions are denoted by $\mathcal{N}(\mathbf{W}, \Omega \otimes \Lambda)$.

The same unconstrained objective of AROW [5] to update the model parameters $\mathbf{W}$ and Σ is adopted and revisited in [4] for the AROMA matrix settings. In the matrix variate normal distribution case, the convex objective becomes

$$\begin{aligned} D_{KL}\left(\mathcal{N}(\mathbf{W}, \Omega \otimes \Lambda) || \mathcal{N}(\mathbf{W}_{t-1}, \Omega_{t-1} \otimes \Lambda_{t-1})\right) \\ + \frac{1}{2r}\left(\max\{0, 1 - \mathbf{u}^T \mathbf{W}(\mathbf{a}^+ - \mathbf{a}^-)\}\right)^2 + \\ \frac{1}{2r}(\mathbf{u}^T \Omega \mathbf{u})((\mathbf{a}^+ - \mathbf{a}^-)^T \Lambda (\mathbf{a}^+ - \mathbf{a}^-)), \end{aligned} \tag{10}$$

where $r > 0$ is a regularization parameter and D_{KL} denotes the *Kullback-Leibler* divergence. The first summand of the objective ensures that the parameters do not change radically from round to round. The middle summand penalizes classification errors. As hinted by the last summand which aims at minimizing the eigenvalues of the confidence matrix $\Sigma = \Omega \otimes \Lambda$, the more examples, the higher the confidence should be.

Denote $\mathbf{a}_t = \mathbf{a}_t^+ - \mathbf{a}_t^-$. The global minimum of the objective (10) was shown to be obtained by the following update rule [4],

$$\mathbf{W}_t = \mathbf{W}_{t-1} + \frac{\max\{0, 1 - \mathbf{u}_t^T \mathbf{W}_{t-1} \mathbf{a}_t\}}{\mathbf{u}_t^T \Lambda_{t-1} \mathbf{u}_t \mathbf{a}_t^T \Omega_{t-1} \mathbf{a}_t + r} \Lambda_{t-1} \mathbf{u}_t \mathbf{a}_t^T \Omega_{t-1}, \tag{11}$$

$$\Omega_t = \Omega_{t-1} - \frac{\mathbf{u}_t^T \Lambda_{t-1} \mathbf{u}_t}{(\mathbf{u}_t^T \Lambda_{t-1} \mathbf{u}_t)(\mathbf{a}_t^T \Omega_{t-1} \mathbf{a}_t) + Kr} \Omega_{t-1} \mathbf{a}_t \mathbf{a}_t^T \Omega_{t-1}, \tag{12}$$

$$\Lambda_t = \Lambda_{t-1} - \frac{\mathbf{a}_t^T \Omega_{t-1} \mathbf{a}_t}{(\mathbf{a}_t^T \Omega_{t-1} \mathbf{a}_t)(\mathbf{u}_t^T \Lambda_{t-1} \mathbf{u}_t) + Lr} \Lambda_{t-1} \mathbf{u}_t \mathbf{u}_t^T \Lambda_{t-1}. \tag{13}$$

5.2 LearnAROMA Mapping

In our case, instead of minimizing the squared-hinge loss ℓ_W^2, we wish to maximize the logarithm of the sigmoid function, namely

$$\ln \frac{1}{1 + e^{-\mathbf{u}^T \mathbf{W} \mathbf{a}}}, \tag{14}$$

or alternatively, we want to minimize

$$\ln\left(1 + e^{-\mathbf{u}^T \mathbf{W} \mathbf{a}}\right), \tag{15}$$

Unfortunately, using (15) as the second summand of the objective (10) does not lead to an analytically solvable update for $\mathbf{W}$. Therefore, we propose to **approximate** (15) by the hinge loss function given by

$$\max\{0, 1 - \mathbf{u}^T \mathbf{W} \mathbf{a}\} \tag{16}$$

rather than the squared-hinge loss used in (10). As shown in Fig. 2, the two losses are similar.

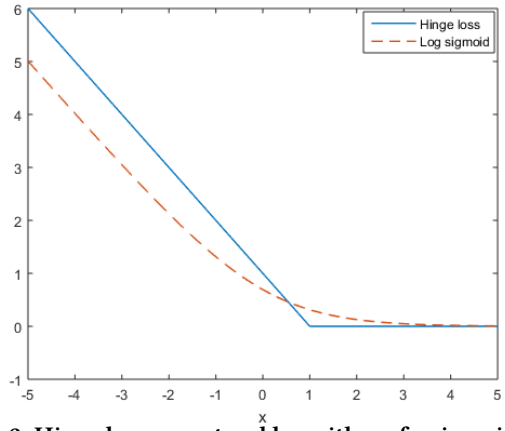

Figure 2: Hinge loss vs. natural logarithm of a sigmoid function.

In this case, the update rule for $\mathbf{W}$ (11) reduces to

$$\mathbf{W}_t = \mathbf{W}_{t-1} + \frac{1}{2r} \Lambda_{t-1} \mathbf{u}_t \mathbf{a}_t^T \Omega_{t-1}, \tag{17}$$

if $\mathbf{u}_t^T \mathbf{W} \mathbf{a}_t < 1$. Otherwise, no update is required. The updates concerning the covariance matrix components (12) and (13) remain unchanged. Our modified AROMA is summarized in Algorithm 3.

Again, once $\mathbf{W}$ is estimated, the resulting item latent vectors are then given by (7).

5.3 Complexity Analysis

LearnMap requires storing the $K \times L$ matrix $\mathbf{W}$ and thus requires a memory of KL. The time complexity of each stochastic gradient descent update is KL as well. LearnAROMA uses a total memory of $KL + K^2 + L^2$ to store the mean matrix $\mathbf{W}$ and the covariance matrices Ω and Λ. The time complexity of each step is $KL + K^2 + L^2$ as well since it involves addition to all elements of the matrices.

Algorithm 3 LearnAROMA

Input:
$S(\mathcal{A}_1)$ - feedback of N users on M_1 mature items
$\mathbf{u}_i$, $1 \le i \le N$ - users latent vectors
$\mathbf{a}_j$, $1 \le j \le M_1$ - mature items attribute vectors
r - regularization parameter
Output:
$\mathbf{W}$ - $K \times L$ attribute-to-feature mapping matrix

repeat
  draw (i, p, q) from $\mathcal{A}_1$
  define $\mathbf{u}_t = \mathbf{u}_i$, $\mathbf{a}_t = \mathbf{a}_p - \mathbf{a}_q$
  if $\mathbf{u}_t^T \mathbf{W} \mathbf{a}_t < 1$ **then**
    update $\mathbf{W}_t$ using (17)
  end if
  update Ω_t using (12)
  update Λ_t using (13)
until convergence

6 EXPEDITING RANDOM EXPLORATION

Consider a new item (cold-start) $j \in \mathcal{A}_2$. Our initial estimate for its latent vector is solely content-based since we do not have any ratings yet for this item and is given by $\hat{\mathbf{v}}_j^0 = \mathbf{W}\mathbf{a}_j$, where $\mathbf{W}$ is obtained using either mapping presented in Section 4 or 5. We now dynamically update the item latent vector estimate as well as its bias with each new rating. We considered estimating the cold-start items bias using content-based mapping, similarly to our latent vector estimation, but this approach did not yield good results.

The BPR-MF criterion is applied on couple of items, one positive and one negative. Therefore, for each new positive feedback, we draw a negative item from the training set for the user who provided the feedback, and vice versa.

For a positive feedback, we denote $j = p$. The optimization criterion is then given by

$$\max_{\hat{\mathbf{v}}_p, \hat{b}_p} \sum_i \ln \frac{1}{1 + e^{-\hat{x}_{ipq}}} - \lambda ||\hat{\mathbf{v}}_p - \hat{\mathbf{v}}_p^0||^2 - \lambda_b \hat{b}_p^2. \tag{18}$$

Here i denotes t exploring users, chosen randomly from $\mathcal{A}_2$, that rated the new item p, and $\hat{x}_{ipq} = \hat{y}_{ip} - \hat{y}_{iq} = \mathbf{u}_i^T \hat{\mathbf{v}}_p - \mathbf{u}_i^T \mathbf{v}_q + \hat{b}_p - b_q$ where q denotes the drawn negative item for the corresponding user, namely it holds that $q \in I_i^-$. The optimization criterion for a negative ranking is obtained by inverting the roles of both items. If necessary, a traditional regularizer is added to avoid overfitting by controlling the norm of the latent vector, that is $\lambda_1 ||\hat{\mathbf{v}}_p||^2$. The resulting DynamicBPR algorithm is shown in Algorithm 4. DynamicBPR is applied to each cold-start item separately, as new interactions with this item are arriving. In the algorithm description, $\mathcal{A}_2(t)$ includes t raws randomly chosen from $\mathcal{A}_2(t)$. Note, that no more than t ratings are used to characterize the new item. In addition, we note that since the users latent vectors are fixed, the optimization is performed only with respect to the items latent vector and is convex. Therefore, convergence is guaranteed.

It is worth mentioning that this online approach does not require retraining the mature model, but only trains the cold-start item latent vectors separately. We also considered updating the mapping matrix along with the items latent vectors, but this method did not yield any significant improvement. This suggests that, given enough mature items and users, the mapping is not drastically changed by new arriving entities.

Algorithm 4 DynamicBPR

Input:
$\hat{\mathbf{v}}_j^0 = \mathbf{W}\mathbf{a}_j$ - new item j initial latent vector, $j \in \mathcal{A}_2$
$\mathcal{A}_2(t)$ - t raws randomly chosen from $\mathcal{A}_2$
$\mathbf{u}_i$, $1 \le i \le N$ - users latent vectors
α, α_b - update step sizes
λ, λ_b - regularization parameter
Output:
$\hat{\mathbf{v}}_j^t$ - new item j estimated latent vector

repeat
  draw a user i from $\mathcal{A}_2(t)$
  if $j \in I_i^+$ **then**
    draw $q \in I_i^-$ from $\mathcal{A}_1$ and set $p = j$
  else
    draw $p \in I_i^+$ from $\mathcal{A}_1$ and set $q = j$
  end if
  $\hat{x}_{ipq} \leftarrow \hat{y}_{ip} - \hat{y}_{iq}$
  if $j \in I_i^+$ **then**
    for $1 \le k \le K$ **do**
      $\hat{v}_{pk} \leftarrow \hat{v}_{pk} + \alpha \left(\frac{e^{-\hat{x}_{ipq}}}{1+e^{-\hat{x}_{ipq}}} u_{ik} - \lambda \left(\hat{v}_{pk} - \hat{v}_{pk}^0 \right) \right)$
    end for
    $\hat{b}_p \leftarrow \hat{b}_p + \alpha_b \left(\frac{e^{-\hat{x}_{ipq}}}{1+e^{-\hat{x}_{ipq}}} - \lambda_b \hat{b}_p \right)$
  else
    for $1 \le k \le K$ **do**
      $\hat{v}_{qk} \leftarrow \hat{v}_{qk} + \alpha \left(-\frac{e^{-\hat{x}_{ipq}}}{1+e^{-\hat{x}_{ipq}}} u_{ik} - \lambda \left(\hat{v}_{qk} - \hat{v}_{qk}^0 \right) \right)$
    end for
    $\hat{b}_q \leftarrow \hat{b}_q + \alpha_b \left(-\frac{e^{-\hat{x}_{ipq}}}{1+e^{-\hat{x}_{ipq}}} - \lambda_b \hat{b}_q \right)$
  end if
until convergence

7 NUMERICAL EVALUATION

This section presents our numerical evaluation and demonstrates the two aspects of our random exploration expedition approach. First, we evaluate the initial estimate of the new items latent vectors using LearnMap and LearnAROMA. Then, we examine the online update of these vectors with DynamicBPR while feedback on the new items is arriving.

7.1 Dataset

We use the MovieLens2K dataset released by HetRec 2011.[1] This dataset provides ratings values which we convert to binary feedback, assuming that users tend to rate movies they have watched [8]. If movie j is rated by user i, then $j \in I_i^+$, namely any rating is considered as positive feedback. If movie j is not rated by user i, then $j \in I_i^-$. We consider $N = 2113$ users and $M = 6590$ movies and randomly split the dataset so that $N_1 = 0.2N$ and $M_2 = 0.2M$ (see Fig. 1); that is, the number of users in the test set correspond to 80% of the total number of users and the cold-start items are 20%

[1] http://grouplens.org/datasets/hetrec-2011

Set	Users	Items	Events
$\mathcal{A}_1$	$N = 2113$	$M_1 = 5328$	845761
$\mathcal{A}_2$	$N_1 = 421$	$M_2 = 1262$	18554
$\mathcal{A}_3$	$N_2 = 1692$	$M_2 = 1262$	95003

Table 1: Dataset split

Method	AUC per user
LearnBPR	0.86029
LearnMap	0.84710
LearnAROMA	0.84989

Table 2: Accuracies for "weak generalization" setting (no cold-start)

Method	AUC per user
LearnMap	0.61123
LearnAROMA	0.62811

Table 3: Accuracies for "strong generalization" setting (cold-start)

of the total number of movies. Table 1 summarizes the dimensions of the resulting model training set $\mathcal{A}_1$, dynamic training set $\mathcal{A}_2$, and test set $\mathcal{A}_3$. In the table, *Events* refer to the number of positive feedback.

The latent space dimension is set to $K = 12$.[2] We consider genres as movies attributes. The attribute vector $\mathbf{a}_j$ are binary vectors of size $L = 20$, with the lth entry equals 1 if the movie exhibits the lth attribute and 0 otherwise. We observed that exploiting additional attributes such as movie directors or actors does not significantly improve the mapping performance, as previously seen in [8]. Therefore, we focus only on genres as attributes.

Our algorithms were implemented using the MyMediaLite[3] recommender system library, presented in [9]. The library is free/open source software, distributed under the terms of the GNU General Public License.

7.2 Performance Metric

Throughout the simulations, we adopt the *area under curve* (AUC) per user, which is a common ranking accuracy measure, as evaluation metric [7]. The AUC per user measures the ratio of pairs correctly ranked with respect to the total number of evaluated pairs in the test set and is defined as

$$\text{AUC} = \sum_{(i,p,q)\in\mathcal{A}_3} \frac{1}{N_2|I_i^+||I_i^-|}\delta(\hat{x}_{ipq}), \tag{19}$$

where $\delta(x) = 1$ if $x \geq 0$ and $\delta(x) = 0$ otherwise.

7.3 BPR-MF Model and Mapping Training

We first consider a scenario deprived of cold-starts, as a baseline. Here, we perform the BPR-MF model training using LearnBPR algorithm (see Section 3.3) on both $\mathcal{A}_1$ and $\mathcal{A}_2$ and use $\mathcal{A}_3$ for testing (i.e., "weak generalization" setting). We then learn the attribute to latent space linear mapping, using either the LearnMap algorithm of [8] (see Section 4) or the LearnAROMA algorithm of Section 5. Both the users and items latent vectors as well as the attribute-to-feature mappings are learnt on the sets $\mathcal{A}_1$ and $\mathcal{A}_2$. We consider both linear and quadratic mappings. However, since the quadratic approach does not significantly improve the ranking performance and add complexity overhead, we only focus on linear mapping.

The latent vectors in LearnBPR are initialized using a Gaussian distribution $\mathcal{N}(0, 0.1)$. The mapping matrix entries are initialized to 0 in both LearnMap and LearnAROMA. In the latter, the covariance matrices $\mathbf{\Omega}$ and $\mathbf{\Lambda}$ are initially set to the $K \times K$ and $L \times L$ identity matrices, respectively. The hyperparameters are chosen via a basic grid search. For LearnBPR, we use $\alpha = \alpha_b = 0.05$, $\lambda = 0.02$ and $\lambda_b = 0$. For LearnMap, we use $\alpha = 0.05$, $\alpha_b = 0.05$, $\lambda = 0.01$ and $\lambda_b = 0.05$. For LearnAROMA, we set $r = 100$.

Table 2 presents the AUC per user of the three algorithms over the test set $\mathcal{A}_3$: LearnBPR, LearnMap and LearnAROMA. Examining Table 2, we observe that the LearnBPR outperforms the two other algorithms that are replacing the items latent vectors $\mathbf{v}_j$ with their mapping-based estimates $\hat{\mathbf{v}}_j = \mathbf{W}\mathbf{a}_j$. This is not surprising since the two mapping based algorithms includes fewer degrees of freedom. Moreover, the results also reveals that the LearnAROMA algorithm is slightly more accurate than the LearnMap algorithm. It is noted that we report these results just to demonstrate that the mapping is well trained using both algorithms and provides recommendations that are only 1.5% less accurate than the "upper bound" accuracy provided by the LearnBPR algorithm.

7.4 Initial Latent Vectors Estimates

We return to our original setting and investigate the performance of the mappings presented in Sections 4 and 5 with respect to new items. Here, we perform the BPR-MF model training and learn the mapping on the set $\mathcal{A}_1$. Then, the cold-starts latent vectors of new items $M_1 < j \leq M$ are computed using (7) (i.e., "strong generalization" setting). Table 3 presents the AUC per user using the initial latent vectors estimates of the new items over the test set $\mathcal{A}_3$.

The results reveal that learning the mapping with the LearnAROMA algorithm provides a 2.7% lift in AUC for new items over the LearnMap algorithm of [8]. Since our approach outperforms LearnMap, it is obviously also better than the baselines presented in [8] (i.e., KNN and linear based methods [8]). We note that random initialization of the latent vectors yields AUC = 0.5 as expected.

7.5 Dynamic Update

Lastly, we demonstrate our approach for expediting random exploration, where we receive a predefined number of exploring users (randomly chosen) interactions with the new items, taken from $\mathcal{A}_2$ (i.e., "strong generalization" setting). The test results calculated over $\mathcal{A}_3$ are presented with respect to the number of available user feedback, either positive or negative, for each item, averaged over $M_2 = 1262$ items. We apply our DynamicBPR algorithm using three

[2] Increasing the latent space dimension did not yield significant improvements on performance.
[3] http://ismll.de/mymedialite

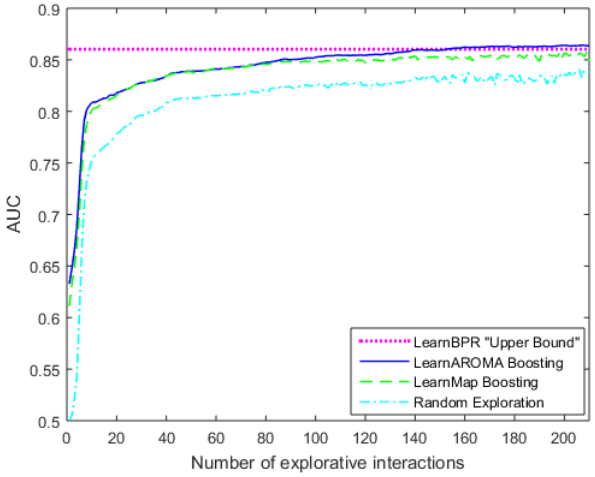

Figure 3: Expediting random exploration using attribute-to-feature mapping

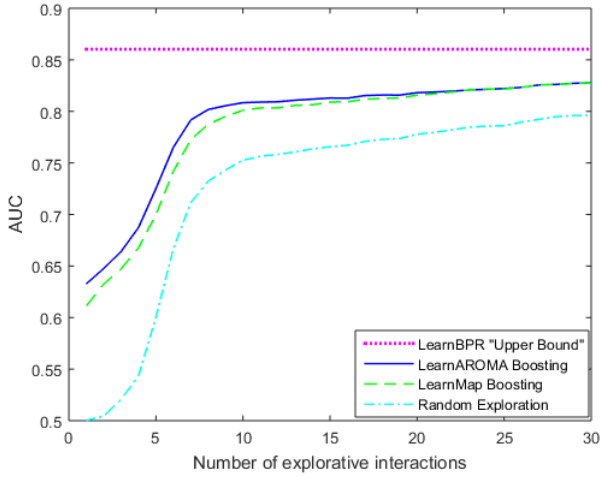

Figure 4: Expediting random exploration using attribute-to-feature mapping (zoom-in).

different initial estimations for the cold-start latent vectors: random $\mathcal{N}(0, 0.1)$, LearnMap (Algorithm 2) and LearnAROMA (Algorithm 3).

Figure 3 presents the AUC accuracies as functions of the number of explorative interactions of all three methods along with the LearnBPR baseline described in Section 7.3 that serves as an "upper bound". We observe that the LearnAROMA based DynamicBPR performance approaches the BPR-MF trained on both $\mathcal{A}_1$ and $\mathcal{A}_2$, namely with no cold-start items. In particular, the LearnAROMA based DynamicBPR reaches 95% accuracy of the LearnBPR "upper bound" using on average 71% less explorative interactions when compared to random exploration (21 vs. 72 interactions). In addition, the LearnAROMA achieves higher accuracy level than that achieved by LearnMap for large number of interactions when exploration reaches saturation and more interactions yield only marginal improvements.

Finally, we focus on the initial phase of the exploration where the benefits of our LearnAROMA based DynamicBPR are more pronounced. Accordingly, Fig. 4 presents a zoom-in of Fig. 3 into the low number of interaction region. Examining the figure, the superiority of our approach over the LearnMap of [8] and especially over random exploration is evident. In addition, we note that LearnAROMA is significantly less sensitive to the choice of hyperparameters in LearnBPR than the LearnMap algorithm. That is, LearnAROMA would yield similar performance even if the model training phase was not performed with optimal hyperparameters, whereas the performance of LearnMap would decrease.

Additional experiment results where we have also updated the mappings individually for each new item with the incoming interactions showed no significant improvements on performance and are thus not presented here.

8 CONCLUDING REMARKS

In this work we consider ways to expedite new item exploration using item attributes in MF-CF based recommenders. In particular, we adopt the approach of [8] where an attribute-to-feature mapping is learnt and used to provide an initial representation for new items, i.e., an initial estimate for their latent vectors. Similarly to [8], we also consider a linear mapping. However, to do so, we adapt the AROMA [4] algorithm, that performs convex optimization for weight matrix estimation in the context of binary classification, to the ranking problem. The resulting algorithm, referred to as LearnAROMA, allows us to learn the attribute to feature mapping. Then, we provide an initial estimate for the new item latent vectors and continue to the exploration phase where users interactions are arriving. To leverage the initial latent vector estimation with the incoming ratings, we present DynamicBPR, a simple update algorithm based on a regularized maximum likelihood convex objective.

Numerical evaluation reveals that our LearnAROMA algorithm provides better initial characterization of the new items than the LearnMAP algorithm of [8] and shows a 2.7% lift in AUC. In addition, we demonstrate the expedition of the exploration phase where our simple update algorithm DynamicBPR achieves comparable accuracies as those provided by a model fully trained with all ratings (an "upper bound" on accuracy) for a rather modest number of exploring users when compared to random exploration. In particular, our algorithm requires on average 71% less exploring users than random exploration to reach a 95% accuracy.

In this work, we consider a random exploration setting where random users are selected to explore the new items (see Section 6). Nevertheless, the benefits of smart exploration, where users who are more likely to interact with the new item are selected for exploration, were recently demonstrated in [6]. In future work, we plan to combine the user selection methods of [6] and the latent vector update algorithm of Section 6 and evaluate the overall lift in performance. Another possible extension of this work is to exploit the estimated confidence matrix along with the mapping, to tune the regularization parameter of DynamicBPR. Indeed, items with attributes for which our mapping has a higher confidence level are likely to have better initial estimate and thus should have a higher regularization term.

REFERENCES

[1] O. Anava, S. Golan, N. Golbandi, Z. Karnin, R. Lempel, O. Rokhlenko, and O. Somekh. 2015. Budget-constrained item cold-start handling in collaborative filtering recommenders via optimal design. *Proc. WWW* (2015).

[2] D. Argawal and B.-C. Chen. 2009. Regression-based Latent Factor Models. *Proc. KDD* (2009).

[3] Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*. 2249–2257.

[4] K. Crammer and G. Chechik. 2012. Adaptive regularization of weight matrices. *Int. Conf. on Machine Learning* (2012).

[5] K. Crammer, A. Kulesza, and M. Dredze. 2009. Adaptive regularization of weight vectors. *NIPS* (2009).

[6] D. Drachsler-Cohen, O. Somekh, S. Golan, M. Aharon, O. Anava, and N. Avigdor-Elgrabli. 2015. ExcUseMe: asking users to help in item cold-start recommendations. *Proc. RecSys* (2015).

[7] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern recognition letters* 27, 8 (2006), 861–874.

[8] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Shmidt-Thieme. 2010. Learning Attribute-to-Feature Mappings for Cold-Start Recommendations. *ICDM'10: Proceedings of the 10th IEEE international conference on data mining* (2010).

[9] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Shmidt-Thieme. 2011. MyMediaLite: a free recommender system library. *Proceeding of the fifth ACM conference on Recommender systems* (2011), 305–308.

[10] A. Gunawardana and C. Meek. 2008. Tied Boltzmann machines for cold starts recommendations. *Proc. RecSys* (2008).

[11] A. Gunawardana and C. Meek. 2009. A unified approach to building hybrid recommender systems. *Proc. RecSys* (2009).

[12] A. K. Gupta and D. K. Nagar. 1999. Matrix variate distributions. *Chapman and Hall/CRC* (1999).

[13] Y. Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. *Proc. KDD* (2008).

[14] Y. Koren and R. Bell. 2010. Advances in collaborative filtering. In *Recommender systems handbook*. Springer, 145–186.

[15] N. Liu, X. Meng, C. Liu, and Q. Yang. 2011. Wisdom of the better few: cold start recommendation via representative based rating elicitation. *Proc. RecSys* (2011).

[16] S.-T. Park and W. Chu. 2009. Pairwise preference regression for cold-start recommendation. *Proc. RecSys* (2009).

[17] S.-T. Park, D. Pennock, O. Madani, N. Good, and D. DeCoste. 2006. Naive filterbots for robust cold-start recommendations. *KDD'06: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining* (2006).

[18] Michael J. Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The adaptive web*. Springer, 325–341.

[19] S. Rendle. 2012. Factorization machines with libFM. *ACM* (2012).

[20] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Shmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. *UAI* (2009).

[21] Steffen Rendle and Lars Schmidt-Thieme. 2008. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*. ACM, 251–258.

[22] R. Salakhutdinov and A. Mnih. 2008. Probabilistic matrix factorization. *NIPS* (2008).

[23] Martin Saveski and Amin Mantrach. 2014. Item cold-start recommendations: learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 89–96.