Bayesian Personalized Ranking for Non-Uniformly Sampled Items

Zeno Gantner
Machine Learning Lab
University of Hildesheim
Germany
gantner@ismll.de

Lucas Drumond
Machine Learning Lab
University of Hildesheim
Germany
ldrumond@ismll.de

Christoph Freudenthaler
Machine Learning Lab
University of Hildesheim
Germany
freudenthaler@ismll.de

Lars Schmidt-Thieme
Machine Learning Lab
University of Hildesheim
Germany
schmidt-thieme@ismll.de

ABSTRACT

In this paper, we describe our approach to track 2 of the KDD Cup 2011. The task was to predict which 3 out of 6 candidate songs were positively rated – instead of not rated at all – by a user. The candidate items were not sampled uniformly, but according to their general popularity.

We develop an adapted version of the Bayesian Personalized Ranking (BPR) optimization criterion [9] that takes the non-uniform sampling of negative test items into account. Furthermore, we present a modified version of the generic BPR learning algorithm that maximizes the new criterion. We use it to train ranking matrix factorization models as components of an ensemble. Additionally, we combine the ranking predictions with rating prediction models to also take into account rating data.

With an ensemble of such combined models, we ranked 8th (out of more than 300 teams) in track 2 of the KDD Cup 2011, without using the additional taxonomic information offered by the competition organizers.

Categories and Subject Descriptors

H3.3 [**Information Search and Retrieval**]: Information Filtering—*Collaborative Filtering*

General Terms

Experimentation, Software

Keywords

matrix factorization, collaborative filtering, personalization, ranking, music recommendation

1. INTRODUCTION

Recommender systems are information systems that learn user preferences from past user actions (ratings, votes, ranked lists, mouse clicks, page views, product purchases, etc.) and suggest items (pages on the web, news articles, jokes, movies, products of any kind, music albums, individual songs, etc.) according to those user preferences.

While rating prediction (*How much will a user like/rate a given item?*) has gained more attention in the recommender systems literature in the past, the task of item prediction (*Which items will a user like/buy?*) (e.g. [3, 7]) is actually more relevant for practical recommender system applications. One approach to item prediction is to treat it as a (per-user) ranking problem, and to suggest the top ranked items.

Bayesian Personalized Ranking (BPR) (section 3.1) is a per-user ranking approach that optimizes a smooth approximation of the area under the ROC curve (AUC) (section 2.3) for each user. BPR has seen successful applications in item prediction from implicit positive-only feedback [9] – even though the framework is not limited to positive-only data – and tag prediction [12].

Models optimized for BPR are suitable when the items to be ranked are sampled uniformly from the set of all items. Yet, this is not always the case, for example when the items to be ranked are sampled according to their general popularity, like in track 2 of the KDD Cup 2011.

To deal with such scenarios, we extend the BPR criterion to a probabilistic ranking criterion that assumes the candidate items (those items that should be ranked by the model) to be sampled from a given distribution (section 3.2). Using this new, more general optimization criterion, we derive an extension of the generic BPR learning algorithm (which is a variant of stochastic gradient ascent) that samples its training examples according to the probability distribution used for the candidate sampling, and thus optimizes the model for the new criterion.

One instance of such a ranking scenario is track 2 of the KDD Cup 2011: There, the task was to predict which 3 out of 6 candidate songs were positively rated (higher than a certain threshold) – instead of not rated at all – by a user. The candidate items were not sampled uniformly, but according to their general popularity, i.e. the number of users who gave a positive rating to them. It turns out that the evaluation criterion, the number of correct predictions, can be approximated by our adapted optimization criterion.

Matrix factorization models are suitable prediction models for recommender systems, and are known to work well for item prediction when trained using the BPR framework.

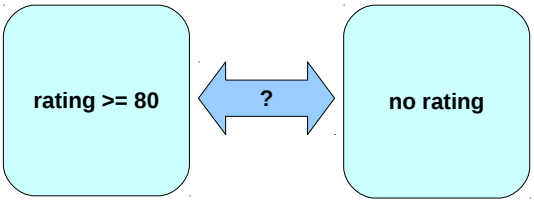

Figure 1: Task of KDD Cup 2011 track 2: Distinguish between tracks a user liked and tracks the user has not rated. Items that have been rated below 80 by the user are not present in the test dataset.

Thus, we used matrix factorization for the KDD Cup (section 3.4).

The main contributions of this article are:

1. We show that for the task of track 2 of the KDD Cup 2011, maximizing the per-user area under the ROC curve (AUC) on the training data is equivalent to minimizing the training error.

2. We adapt the Bayesian Personalized Ranking (BPR) framework, which relies on maximizing a smooth approximation of AUC, to scenarios where negative items are sampled non-uniformly.

3. We develop a scalable matrix factorization model that is trained using the adapted BPR framework.

4. We integrate rating information into the predictions with two different multiplicative schemes, combining the probability of a rating happening with the estimated rating, and the probability of a rating of 80 or more, respectively.

We use the highly competitive KDD Cup 2011 to demonstrate the suitability of our approach.

2. PROBLEM STATEMENT

The task of track 2 of the 2011 KDD Cup was to predict which 3 tracks out of 6 candidates a user will like – rate with a score of 80 or higher – for a set of users, given the past ratings of a superset of the users. Additionally, an item taxonomy expressing relations between tracks, albums, artists, and genres was provided by the contest organizers. We did not use this additional data in our approach.

The 3 candidate tracks that have not been rated highly by the user have not been rated at all by the user. They were not sampled uniformly, but according to how often they are rated highly in the overall dataset.

To put it briefly, the task was to distinguish items (in this case tracks) that were likely to be rated with a score of 80 or higher by the user from items that were generally popular, but not rated by the user (Figure 1). This is similar to the task of distinguishing the highly rated items from all other generally popular ones, which we call the "liked" contrast (Figure 2).

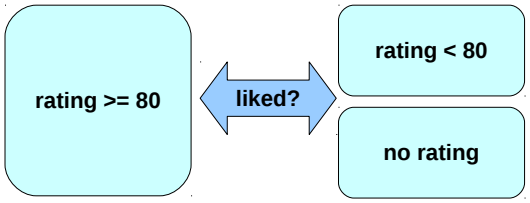

Figure 2: The "liked" contrast: We say that a user likes an item if they rated it with 80 or higher.

2.1 Notation

Scalar variables are set in the default math font, e.g. a, b, c, while matrices (upper case) and vectors (lower case) are in bold face, e.g. $\mathbf{A}, \mathbf{B}, \mathbf{x}, \mathbf{y}$.

Be $\mathcal{U}$ the set of users and $\mathcal{I}$ the set of items (tracks, albums, artists, genres). Without loss of generality, we will refer to users and items using natural numbers between 1 and $|\mathcal{U}|$ or $|\mathcal{I}|$, respectively.

Generally, the training and testing sets in track 2 of the KDD Cup 2011 have the following structure:

$$\mathcal{D}^{\text{train}} \subset \mathcal{U} \times \mathcal{I} \times [0, 100] \quad (1)$$

$$\mathcal{D}^{\text{test}} \subset \mathcal{U} \times \mathcal{I} \times \{0, 1\}. \quad (2)$$

with $\forall (u, i, p_{u,i}) \in \mathcal{D}^{\text{test}} : \neg\exists (u, i, r_{u,i}) \in \mathcal{D}^{train}$. The training set $\mathcal{D}^{\text{train}}$ contains ratings, and the testing set $\mathcal{D}^{\text{test}}$ contains binary variables that represent whether a user has rated an item with a score of at least 80 or not.

For convenience, we use $\mathcal{I}_u^+$ for positive items and $\mathcal{I}_u^-$ for negative similarly to the notation in Rendle et al. [9]. Depending on the context, $\mathcal{I}_u^+$ and $\mathcal{I}_u^-$ may refer to the positive and negative items in the training or test set. What determines whether an item is positive or negative may differ (see below section 2.2).

We will use the letter p for assignments to $0, 1$ or their probabilities, and s for arbitrary scores $\in \mathbb{R}$. $p_{u,i}$ says whether item i was rated (highly) by user u. $\hat{p}_{u,i}(\Theta)$, usually simplified to $\hat{p}_{u,i}$, is the decision (estimation) of a model Θ for the true assignment $p_{u,i}$. Output scores $\hat{s}_{u,i}(\Theta) = \hat{s}_{u,i}$ refer to arbitrary numerical predictions of recommendation models Θ, where higher scores refer to higher positions in the ranking. Such estimated rankings can then be used to make decisions $\hat{p}_{u,i}$.

2.2 Contrasts

Depending on the exact contrast we wish to learn, there are certain different conditions for what is in the set of positive ($\mathcal{I}_u^+$) and negative ($\mathcal{I}_u^-$) items for each user.

Track 2 Contrast. The contrast to be learned for the KDD Cup 2011 ignores all ratings below a score of 80: Such ratings are not used for sampling the negative candidate items – only items that are not rated by to users are potential candidates (Figure 1).

$$\mathcal{I}_u^{+(\text{t2})} := \{i | \exists r_{u,i} \geq 80 : (u, i, r_{u,i}) \in \mathcal{D}^{\text{train}}\} \quad (3)$$

$$\mathcal{I}_u^{-(\text{t2})} := |\mathcal{I}| - \{i | \exists r_{u,i} : (u, i, r_{u,i}) \in \mathcal{D}^{\text{train}}\} \quad (4)$$

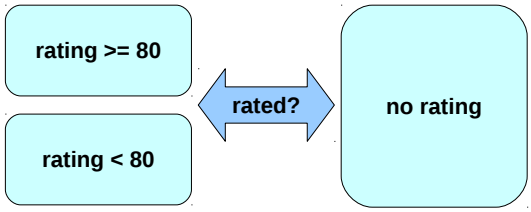

Figure 3: The "rated" contrast: The question is not *how* a user has rated an item, but *if*.

Note that all items i with $r_{u,i} < 80$ do not belong to either of the two sets.

"Liked" Contrast. The "liked" contrast differentiates between what users have rated highly (80 or more), and what they have not rated or rated with a score below 80 (Figure 2):

$$\mathcal{I}_u^{+(\text{liked})} := \{i | \exists r_{u,i} \geq 80 : (u,i,r_{u,i}) \in \mathcal{D}^{\text{train}}\} \quad (5)$$

$$\mathcal{I}_u^{-(\text{liked})} := \mathcal{I} - \mathcal{I}_u^{+(\text{liked})} \quad (6)$$

As can easily be seen from the definition of $\mathcal{I}_u^{-(\text{liked})}$, the split between positive and negative items is exhaustive for each user.

"Rated" Contrast. Finally, the "rated" contrast differentiates what users have rated vs. not rated (Figure 3):

$$\mathcal{I}_u^{+(\text{rated})} := \{i | \exists r_{u,i} : (u,i,r_{u,i}) \in \mathcal{D}^{\text{train}}\} \quad (7)$$

$$\mathcal{I}_u^{-(\text{rated})} := \mathcal{I} - \mathcal{I}_u^{+(\text{rated})} \quad (8)$$

Again, this split is exhaustive for each user.

2.3 Error Measure

The evaluation criterion is the error rate, which is just the relative number of wrong predictions:

$$e = 1 - \frac{1}{|\mathcal{D}^{test}|} \sum_{(u,i,p_{u,i})} \delta(p_{u,i} = \hat{p}_{u,i}), \quad (9)$$

where $\delta(x = y)$ is 1 if the condition (in this case: $x = y$) holds, and 0 otherwise, and $\hat{p}_{u,i}$ is the prediction whether item i is rated 80 or higher by user u. For a single user, the error rate is

$$e_u = 1 - \frac{1}{|\mathcal{I}_u^+| + |\mathcal{I}_u^-|} \sum_{(u,i,p_{u,i})} \delta(p_{u,i} = \hat{p}_{u,i}). \quad (10)$$

For the KDD Cup 2011, we have the additional constraints that for every highly rated item of each user, there is an item that has not been rated in the evaluation set $\mathcal{D}^{test}$, and that exactly half of the candidate items must be given a prediction of $\hat{p}_{u,i} = 1$. We call this the *1-vs.-1 evaluation scheme*.

The area under the ROC curve (AUC) is a ranking measure that can also be computed for the KDD Cup scenario. The per-user AUC on the test set can be defined as follows:

$$AUC(u) = \frac{1}{|\mathcal{I}_u^+||\mathcal{I}_u^-|} \sum_{i \in \mathcal{I}_u^+} \sum_{j \in \mathcal{I}_u^-} \delta(\hat{p}_{u,i} > \hat{p}_{u,j}) \quad (11)$$

1: **procedure** LEARNBPR($\mathcal{D}^{\text{train}}, \alpha, \lambda$)
2: initialize $\hat{\Theta}$
3: **repeat**
4: draw (u,i) from $\mathcal{D}^{\text{train}}$
5: draw j uniformly from $\mathcal{I}_u^-$
6: $\hat{\Theta} \leftarrow \hat{\Theta} + \alpha \left(\frac{e^{-\hat{s}_{u,i,j}}}{1+e^{-\hat{s}_{u,i,j}}} \cdot \frac{\partial}{\partial \hat{\Theta}} \hat{s}_{u,i,j} - \lambda \cdot \hat{\Theta} \right)$
7: **until** convergence
8: **return** $\hat{\Theta}$
9: **end procedure**

Figure 4: LearnBPR: Optimizing BPR-Opt using stochastic gradient ascent.

1: **procedure** LEARNWBPR($\mathcal{D}^{\text{train}}, \alpha, \lambda$)
2: initialize $\hat{\Theta}$
3: **repeat**
4: draw (u,i) from $\mathcal{D}^{\text{train}}$
5: draw j from $\mathcal{I}_u^-$ proportionally to w_j
6: $\hat{\Theta} \leftarrow \hat{\Theta} + \alpha \left(\frac{e^{-\hat{s}_{u,i,j}}}{1+e^{-\hat{s}_{u,i,j}}} \cdot \frac{\partial}{\partial \hat{\Theta}} \hat{s}_{u,i,j} - \lambda \cdot \hat{\Theta} \right)$
7: **until** convergence
8: **return** $\hat{\Theta}$
9: **end procedure**

Figure 5: LearnWBPR: Optimizing WBPR using stochastic gradient ascent. The difference to LearnBPR is the sampling in line 5.

The average AUC over all relevant users is

$$AUC = \frac{1}{|U^{\text{test}}|} \sum_{u \in U^{\text{test}}} AUC(u), \quad (12)$$

where $U^{test} = \{u | (u,i,p_{u,i}) \in \mathcal{D}^{\text{test}}\}$ is the set of users that are taken into account in the evaluation.

LEMMA 1. *In the 1-vs.-1 evaluation scheme the per-user accuracy $1 - e_u$ grows strictly monotonically given the per-user area under the ROC curve (AUC) and vice versa.*

PROOF. Items are ordered according to their scores $\hat{s}_{u,i}$. Be n_{tp} and n_{tn} the number of true positives and true negatives, respectively. Given $\mathcal{I}_u^+$, $\mathcal{I}_u^-$, $\text{AUC}(u) = \frac{n_{tp} \cdot n_{tn}}{|\mathcal{I}_u^+||\mathcal{I}_u^-|} < 1$ and $1 - e_u = \frac{n_{tp}+n_{tn}}{|\mathcal{I}_u^+|+|\mathcal{I}_u^-|} < 1$. If the scores change s.t. $\hat{p}'_{u,i} \neq \hat{p}_{u,i}$ for exactly two items that have been wrongly classified before, then $\text{AUC}'(u) = \frac{(n_{tp}+1)\cdot(n_{tn}+1)}{|\mathcal{I}_u^+||\mathcal{I}_u^-|} > \text{AUC}(u)$ and $1 - e'_u = \frac{n_{tp}+1+n_{tn}+1}{|\mathcal{I}_u^+|+|\mathcal{I}_u^-|} > 1 - e_u$. □

This result can be easily extended from the user-specific evaluation to the overall evaluation:

LEMMA 2 (COROLLARY). *In the 1-vs.-1 scheme the overall accuracy $1 - e$ grows strictly monotonically, given the AUC averaged over all users, and vice versa.*

A similar proof to the one above can be constructed for this corollary.

This means that maximizing the AUC on the training data (while preventing overfitting) is a viable strategy for learning models that perform well under 1-vs.-1 evaluation scheme.

3. METHODS

One approach to solve the task of track 2 of KDD Cup 2011 is to assign scores to the 6 candidate items of each user, and then to pick the 3 highest-scoring candidates. This is similar to classical top-N item recommendation. The decision function is

$$\hat{p}_{u,i} = \begin{cases} 1, & |\{j|(u,j) \in \mathcal{D}^{\text{test}} \wedge \hat{s}_{u,i} > \hat{s}_{u,j}\}| \geq 3 \\ 0, & \text{else} \end{cases}. \quad (13)$$

3.1 Bayesian Personalized Ranking (BPR)

The Bayesian Personalized Ranking (BPR) framework [9] consists of an optimization criterion and a gradient-based learning algorithm for personalized item recommendations. BPR is based on the idea of reducing ranking to pairwise classification [1].

The BPR optimization criterion is

$$\text{BPR-OPT} = \max_{\Theta} \sum_{(u,i,j) \in \mathcal{D}_S} \ln \sigma(\hat{s}_{u,i,j}) - \lambda \|\Theta\|^2, \quad (14)$$

where $\mathcal{D}_S \in \bigcup_{u \in \mathcal{U}} \mathcal{I}_u^+ \times \mathcal{I}_u^-$ are the item pairs to sample from, $\sigma(x) = \frac{1}{1+e^{-x}}$ is the logistic function, $\hat{s}_{u,i,j}$ is the pairwise prediction for user u and items i, j, and $\lambda\|\Theta\|^2$ is a regularization term to prevent overfitting. Because the size of the training data $\mathcal{D}_S$ is quadratic in the number of items, the BPR learning algorithm (Figure 4) samples from $\mathcal{D}_S$ instead of going over the complete set of item pairs.

Note that maximizing BPR-OPT is similar to maximizing the AUC (eq. 12), by approximating the non-differentiable $\delta(x)$ by the differentiable sigmoid function $\sigma(x)$. See [9] for a more detailed explanation.

3.1.1 BPR and CofiRank

BPR is similar in spirit to CofiRank [13]. BPR uses stochastic gradient ascent for learning. CofiRank has a learning method that alternates between optimizing the user and item factors with bundle methods. Both have ranking losses – pairwise logistic loss vs. pairwise soft-margin loss. There is also a variant of CofiRank that is optimized for normalized discounted cumulative gain (NDCG). While both approaches can be used in principle to handle any kind of ranking, BPR has been used for "ranking" positive-only feedback (observed vs. missing/not observed), and CofiRank has been used for ordinal ranking (ranking of known ratings). The trick used by Weimer et al. [14] for computing the ranking loss in $O(n \log n)$ instead of $O(n^2)$ (where n is the length of the ranked list) is particularly useful for ordinal ranking, where n is seldom as large as $|\mathcal{I}|$, and for exact (as a opposed to stochastic) optimization, where the complete loss/gradient has to be computed. Still, we will look at ways to exploit this trick for our method in the future.

3.2 Weighted BPR (WBPR)

Because the negative items in the training data are all weighted identically – and not according to their global popularity – optimizing a predictive model for BPR-OPT does not lead to good results in track 2 of the KDD Cup 2011.

The sampling probability for an item is proportional to

$$w_j = \frac{1}{|\mathcal{D}|} \sum_{u \in \mathcal{U}} \delta(j \in \mathcal{I}_u^+). \quad (15)$$

Taking into account the sampling probability of the negative items, the modified optimization criterion is

$$\text{WBPR} = \max_{\Theta} \sum_{(u,i,j) \in \mathcal{D}_S} w_u w_i w_j \ln \sigma(\hat{s}_{u,i,j}) - \lambda \|\Theta\|^2, \quad (16)$$

where $w_u = \frac{1}{|\mathcal{I}_u^+|}$ is a weight that balances the contribution of each user to the criterion, and $w_i = 1$ (positive items are sampled uniformly).

Note that WBPR is not limited to the task of the KDD Cup 2011: The weights w_u, w_i, w_j can be adapted to other scenarios (sampling probabilities).

3.3 Learning by Adapted Sampling

To train a model according to the modified optimization criterion, we adapted the original learning algorithm (Figure 4); instead of sampling negative items uniformly, we sample them according to their overall popularity w_j (line 5 in Figure 5).

3.4 Matrix Factorization Optimized for WBPR

The pairwise prediction $\hat{s}_{u,i,j}$ is often expressed as the difference of two single predictions:

$$\hat{s}_{u,i,j} := \hat{s}_{u,i} - \hat{s}_{u,j}. \quad (17)$$

We use the BPR framework and its adapted sampling extension to learn matrix factorization models with item biases:

$$\hat{s}_{u,i} := b_i + \langle \mathbf{w}_u, \mathbf{h}_i \rangle, \quad (18)$$

where $b_i \in \mathbb{R}$ is a bias value for item i, $\mathbf{w}_u \in \mathbb{R}^k$ is the latent factor vector representing the preferences of user u, and $\mathbf{h}_i \in \mathbb{R}^k$ is the latent factor vector representing item i.

The optimization problem is then

$$\max_{\mathbf{W},\mathbf{H},\mathbf{b}} \sum_{(u,i,j) \in \mathcal{D}_S} w_u w_i w_j \ln \sigma(b_i - b_j + \langle \mathbf{w}_u, \mathbf{h}_i - \mathbf{h}_j \rangle) - \lambda_u \|\mathbf{W}\|^2 - \lambda_i \|\mathbf{H}\|^2 - \lambda_b \|\mathbf{b}\|^2. \quad (19)$$

The training algorithm LearnWBPR-MF (Figure 6) (approximately) optimizes this problem using stochastic gradient ascent. It is an instance of the generic LearnWBPR algorithm. The parameter updates make use of the partial derivatives of the local error with respect to the current parameter. The matrix entries must be initialized to non-zero values because otherwise all gradients and regularization updates for them would be zero, and thus no learning would take place. The item bias vector $\mathbf{b}$ does not have this problem. Note that the λ constants in the learning algorithm are not exactly equivalent to their counterparts in the optimization criterion. We also have two different regularization constants λ_i and λ_j which lead to different regularization updates for positive and negative items.

3.5 Ensembles

To get more accurate predictions, we trained models for different numbers of factors k and using different regularization settings. We combined the results of the different models, and of the same models at different training stages.

We used two different combination schemes, score averaging and vote averaging.

Score Averaging. If models have similar output ranges, for example the same model at different training stages, we can

1: **procedure** LEARNWBPR-MF($\mathcal{D}^{\text{train}}, \alpha, \lambda_u, \lambda_i, \lambda_j, \lambda_b$)
2: set entries of $\mathbf{W}$ and $\mathbf{H}$ to small random values
3: $\mathbf{b} \leftarrow \mathbf{0}$
4: **repeat**
5: draw (u, i) from $\mathcal{D}^{\text{train}}$
6: draw j from $\mathcal{I}_u^-$ proportionally to w_j
7: $\hat{s}_{u,i,j} \leftarrow b_i - b_j + \langle \mathbf{w}_u, \mathbf{h}_i - \mathbf{h}_j \rangle$
8: $x \leftarrow \frac{e^{-\hat{s}_{u,i,j}}}{1+e^{-\hat{s}_{u,i,j}}}$
9: $b_i \leftarrow b_i + \alpha\,(x - \lambda_b b_i)$
10: $b_j \leftarrow b_j + \alpha\,(-x - \lambda_b b_j)$
11: $\mathbf{w}_u \leftarrow \mathbf{w}_u + \alpha\,(x \cdot (\mathbf{h}_i - \mathbf{h}_j) - \lambda_u \mathbf{w}_u)$
12: $\mathbf{h}_i \leftarrow \mathbf{h}_i + \alpha\,(x \cdot \mathbf{w}_u - \lambda_i \mathbf{h}_i)$
13: $\mathbf{h}_j \leftarrow \mathbf{h}_j + \alpha\,(x \cdot (-\mathbf{w}_u) - \lambda_j \mathbf{h}_j)$
14: **until** convergence
15: **return** $\mathbf{W}, \mathbf{H}, \mathbf{b}$
16: **end procedure**

Figure 6: Optimizing a matrix factorization model for WBPR using stochastic gradient ascent.

achieve more accurate predictions by averaging the scores predicted by the models:

$$\hat{s}_{u,i}^{\text{score-ens}} = \sum_m \hat{s}_{u,i}^{(m)}. \tag{20}$$

Vote Averaging. If we do not know whether the scale of the scores is comparable, we can still average the voting decisions of different models:

$$\hat{s}_{u,i}^{\text{vote-ens}} = \sum_m \hat{p}_{u,i}^{(m)}. \tag{21}$$

Other possible combination schemes would be ranking ensembles [11], and of course weighted variants of all schemes discussed here.

Greedy Forward Selection of Models. Because selecting the optimal set of models to use in an ensemble is not feasible if the number of models is high, we perform a greedy forward search to find a good set of ensemble components: This search procedure tries all candidate components sorted by their validation set accuracy, and adds the candidate to the ensemble if it improves the current mix. When searching a large number (e.g. $> 2,000$) of models, we ignored candidates above a given error threshold.

3.6 Integrating Rating Information

Except for the rating threshold of 80, the methods presented so far in this paper do not take into account the actual rating values. We suggest two different schemes of combining probabilities of whether an item has been rated by a user with rating predictions produced by a matrix factorization model that incorporates user and item biases (see e.g. [8, 10]:

$$\begin{aligned}\min_{\mathbf{W},\mathbf{H},\mathbf{b}^{\text{U}},\mathbf{b}^{\text{I}}} \sum_{(u,i,r_{u,i})\in\mathcal{D}^{\text{train}}} & (\sigma(\mu + b_u^{\text{U}} + b_i^{\text{I}} + \langle \mathbf{w_u}, \mathbf{h_i} \rangle) - r_{u,i})^2 \\ & + \lambda_b(\|\mathbf{b}^{\text{U}}\|^2 + \|\mathbf{b}^{\text{I}}\|^2) \\ & + \lambda_u \|\mathbf{W}\|^2 + \lambda_i \|\mathbf{H}\|^2,\end{aligned} \tag{22}$$

where μ is the global rating average. The model is trained

1: **procedure** CREATEDATASET($\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{test}}, n$)
2: $\mathcal{D}^{\text{train-val}} \leftarrow \mathcal{D}^{\text{train}}$
3: $\mathcal{U}^{\text{test}} \leftarrow \{u | (u, i, p_{u,i}) \in \mathcal{D}^{\text{test}}\}$
4: **for all** $u \in \mathcal{U}^{\text{test}}$ **do**
5: $I^+ \leftarrow \{n$ random items from $\mathcal{I}_u^{+(\text{t2})}\}$
6: $\mathcal{D}^{\text{test-val}} \leftarrow \mathcal{D}^{\text{test-val}} \dot\cup \{u\} \times I^+ \times \{1\}$
7: $I^- \leftarrow \{n$ items from $\mathcal{I}_u^{-(\text{t2})}$ sampled prop. to popul.$\}$
8: $\mathcal{D}^{\text{test-val}} \leftarrow \mathcal{D}^{\text{test-val}} \dot\cup \{u\} \times I^- \times \{0\}$
9: **for all** $i \in I^+ \dot\cup I^-$ **do**
10: $\mathcal{D}^{\text{train-val}} \leftarrow \mathcal{D}^{\text{train-val}} - \{(u, i, r_{u,i})\}$
11: **end for**
12: **end for**
13: **return** $(D^{\text{train-val}}, D^{\text{test-val}})$
14: **end procedure**

Figure 7: Sampling procedure for the validation split.

using stochastic gradient descent with the bold-driver heuristic that dynamically adapts the learn rate. Using this heuristic for learning matrix factorizations was suggested by Gemulla et al. [6].

Estimating Probabilities. First, we describe how we compute probabilities from prediction scores of models that were trained to decide whether an item has been rated or not (Figure 3).

$$\hat{p}_{u,i}^{\text{rated}} = \sum_{k=1}^{5} \sum_{l=k+1}^{5} \sum_{m=l+1}^{5} \sigma(\hat{s}_{u,i,j_k}^{\text{rated}}) \sigma(\hat{s}_{u,i,j_l}^{\text{rated}}) \sigma(\hat{s}_{u,i,j_m}^{\text{rated}}), \tag{23}$$

where $\hat{s}_{u,i,j_1}^{\text{rated}} \dots \hat{s}_{u,i,j_5}^{\text{rated}}$ refer to the score estimates of the other 5 candidates. Note that the models for $\hat{s}^{\text{rated}}$ are trained using all ratings as input, not just the ones of 80 or higher. The intuition behind this way of probability estimation is as follows: $\sigma(\hat{s}_{u,i,j_k}^{\text{rated}}) \in (0, 1)$ can be interpreted, similar to the case of logistic regression (e.g. [2]) as the probability that item i is ranked higher (more likely to be rated) than item j by user u. We know that exactly 3 items are rated by the user, which means we need to estimate how probable it is that a given item is ranked higher than 3 other items. Eq. 23 sums up the probabilities for the different cases where this holds.

Scheme 1: Multiply with Rating Prediction. The first scheme takes a "rated" probability and multiplies it with a rating prediction from a model trained on the rating data:

$$\hat{s}_{u,i}^{\text{one}} = \hat{p}_{u,i}^{\text{rated}} \cdot \hat{r}_{u,i}, \tag{24}$$

where $\hat{r}_{u,i}$ is the predicted rating.

Scheme 2: Multiply with Rating Probability. The second scheme takes a "rated" probability and multiplies it with the probability that the item, if rated, gets a rating of 80 or more by the user:

$$\hat{s}_{u,i}^{\text{two}} = \hat{p}_{u,i}^{\text{rated}} \cdot \hat{p}_{u,i}^{\geq 80}, \tag{25}$$

where $\hat{p}_{u,i}^{\geq 80}$ is the estimated probability of $r_{u,i} \geq 80$. We estimate $\hat{p}_{u,i}^{\geq 80}$ using several different rating prediction models:

$$\hat{p}_{u,i}^{\geq 80} = \sum_k \delta(\hat{r}_{u,i}^{(k)} \geq 80). \tag{26}$$

	Ratings	
	validation split	competition split
# users	249,012	249,012
# items	296,111	296,111
# ratings	61,640,890	61,944,406
# sparsity	0.999164	0.9991599
# test users	101,172	101,172
# test items	128,114	118,363
	Ratings ≥ 80	
	validation split	competition split
# users	248,502	248,529
# items	289,234	289,303
# ratings	22,395,798	22,699,314
# sparsity	0.9996884	0.9996843
# test users	101,172	101,172
# test items	128,114	118,363

Table 1: Characteristics of the validation and competition splits when considering all ratings (Figure 3) and the ratings of 80 or more (Figure 2), respectively.

4. EXPERIMENTS

4.1 Datasets

The *Yahoo! Music* dataset used for KDD Cup 2011 is described in [4]. We created a validation split from the training set so that we could estimate the accuracy of different models, and use those estimates to drive the composition of ensemble models. The procedure to create the split, based on the task description of track 2[1], is described in Figure 7. In the case of the KDD Cup data, the number of positive items per user in the test set is $n = 3$. Table 1 shows the characteristics of the datasets.

4.2 Rating Prediction

Table 2 contains the rating prediction accuracy in terms of root mean square error (RMSE) and mean absolute error (MAE) on the validation split for different hyperparameter combinations. For the original split, the true rating values are not (yet) available, so we do not report them.

4.3 Track 2 Results

We trained all models on both splits. Some results for the validation splits, and from the leaderboard are in Table 3. We did not submit all of those models to the competition, so some leaderboard results are missing.

4.4 Final Submission

For our final submission (see Table 3), we used the second rating integration scheme (eq. 25). To estimate $\hat{p}_{u,i}^{\text{rated}}$, we created a score ensemble (section 3.5) from candidate models described in Table 4, with a candidate error threshold of 5.2% – models with a higher validation error were not considered for the ensemble. We estimated the probabilities for a high rating $\hat{p}_{u,i}^{\geq 80}$ according to eq. 26, from the models listed in Table 5.

[1] `http://kddcup.yahoo.com/datasets.php`

4.5 Reproducibility

The algorithms described here are part of the MyMediaLite[2] recommender system library [5]. `http://ismll.de/mymedialite/examples/kddcup2011.html` describes all steps needed to reproduce the experiments presented in this paper.

5. SUMMARY AND OUTLOOK

We described how to adapt the Bayesian Personalized Ranking (BPR) framework to scenarios where negative items are sampled non-uniformly, like in track 2 of the KDD Cup 2011, yielding the optimization criterion WBPR. This criterion can for instance be used to learn scalable matrix factorization models, which we used for our participation in the KDD Cup 2011. In addition to ensembles of different WBPR matrix factorization models, we enhanced the predictions by integrating additional rating information. The experiments presented in this paper, and the ranking on the KDD Cup leaderboard[3], suggest that our methods are suitable for such recommendation tasks.

This article is merely a description of the methods we developed and used for this competition. There are several aspects worth further investigation.

First of all, we reduce a classification problem (optimization for the error rate) to a ranking problem, which we again solve using a reduction to pairwise classification. While in general item recommendation scenarios ranking is the problem we want to solve, it would be still interesting to see whether improvements are possible by directly training a classifier.

We have not used the item taxonomy, so a next step will be to make use of this additional information, as well as trying other ways of integrating the rating information (see section 3.6). A fully Bayesian treatment of the WBPR framework, i.e. by estimating parameter distributions, could yield models that have less hyperparameters, while having accuracies comparable to ensembles of the current models.

For the competition, we performed all training on the "liked" (Figure 2) and "rated" (Figure 3) contrasts, but not on the proper contrast (Figure 1) that was used for evaluation the KDD Cup. We will investigate the benefits of learning that correct contrast. Using a soft-margin loss instead of a logistic loss could allow us to speed up learning (see section 3.1.1). Finally, we are looking forward to see the other contributions to the KDD Cup 2011, and to find out which aspects and details made a real difference in the competition, and which insights gained can be utilized for other recommendation tasks.

6. ACKNOWLEDGMENTS

We thank the reviewer(s) for their helpful feedback. This work was funded in part by the German Research Council (Deutsche Forschungsgemeinschaft, DFG) within the project *Multirelational Factorization Models*. The development of the MyMediaLite software was co-funded by the European Commission FP7 project *MyMedia* under the grant agreement no. 215006.

[2] `http://ismll.de/mymedialite`

[3] Note that we did not make use of the additional hierarchical information.

Model	Hyperparameters	RMSE	MAE
MF	$k = 40, \lambda_u = 2.3, \lambda_i = 1.4, \lambda_b = 0.009, \alpha = 0.00002, i = 30$	25.37	16.88
MF	$k = 60, \lambda_u = 3.9, \lambda_i = 1.7, \lambda_b = 0.00005, \alpha = 0.00005, i = 55$	25.35	16.67

Table 2: Rating prediction accuracy on the validation split for different matrix factorization models (eq. 22).

Model	Hyperparameters	Validation	Leaderboard
most popular	–	29.8027	42.8546
most rated	–	29.0802	
WR-MF [7]	$k = 60, \lambda = .0001, c_{\text{pos}} = 320, i = 30$		13.7587
WBPR-MF ("liked" contrast)	$k = 240, \lambda_u = .01, \lambda_i = .005, \lambda_j = .0005, \lambda_b = .0000001, i = 222$	6.275	6.0449
WBPR-MF ("rated" contrast)	$k = 320, \lambda_u = .0075, \lambda_i = .005, \lambda_j = .00025, \lambda_b = .000015, i = 53$	5.4103	6.0819
final submission	see section 4.4	3.80178	4.4929

Table 3: Validation set and KDD Cup 2011 leaderboard error percentages for different models. i refers to the number of iterations used to train the model. See the method section for details about the methods.

k	λ_u	λ_i	λ_j	λ_b	α	i	#
480	0.005	$\{0.0015, 0.0025, 0.0035\}$	$\{0.00015, 0.00025, 0.00035\}$	$\{0.000015, 0.00002\}$	0.04	$\{10, \ldots, 200\}$	3,420

Table 4: Candidate components of the score ensemble used for estimating $\hat{p}_{u,i}^{\mathbf{rated}}$ (section 3.5). The last column shows the number of different models resulting from combining the hyperparameter values in that row.

k	λ_u	λ_i	λ_b	α	i	#
40	$\{1.9, 2.0, 2.2\}$	$\{0.8, 1.0, 1.2\}$	$\{0.000075, 0.0001, 0.0075\}$	0.00002	$\{8, \ldots 11, 20, 24, 30, 31, 33, 38, \ldots, 41\}$	351
40	$\{2.1, 2.3\}$	$\{1.1, 1.4\}$	$\{0.006, 0.0075, 0.009\}$	0.00002	$\{8, \ldots 11, 20, 24, 30, 31, 33, 38, \ldots, 41\}$	156
60	$\{3, 3.5\}$	$\{1.1, 1.25, 1.5\}$	$\{0.0000075, 0.00005\}$	0.00005	$\{30, 50, 70, 89, \ldots, 93\}$	84
60	$\{3.4, 3.9\}$	$\{1.2, 1.5, 1.7\}$	$\{0.00005\}$	0.00005	$\{30, 50, 70, 89, \ldots, 93\}$	48

Table 5: Rating prediction models used for estimating $\hat{p}_{u,i}^{\geq 80}$ (eq. 26) in the final KDD Cup submission. The last column shows the number of different models resulting from combining the hyperparameter values in that row.

7. REFERENCES

[1] M.-F. Balcan, N. Bansal, A. Beygelzimer, D. Coppersmith, J. Langford, and G. B. Sorkin. Robust reductions from ranking to classification. *Machine Learning*, 72(1-2):139–153, 2008.

[2] C. M. Bishop. *Pattern recognition and machine learning*. Springer, New York, 2006.

[3] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems. Springer-Verlag*, 22/1, 2004.

[4] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer. The Yahoo! Music Dataset and KDD-Cup'11. In *KDD-Cup Workshop*, 2011.

[5] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. MyMediaLite: A free recommender system library. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys 2011)*, 2011.

[6] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *KDD*, 2011.

[7] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM 2008*, 2008.

[8] Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8), 2009.

[9] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI 2009*, 2009.

[10] S. Rendle and L. Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *RecSys '08: Proceedings of the 2008 ACM Conference on Recommender Systems*. ACM, 2008.

[11] S. Rendle and L. Schmidt-Thieme. Factor models for tag recommendation in BibSonomy. In *Discovery Challenge at ECML PKDD 2009*, 2009.

[12] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In B. D. Davison, T. Suel, N. Craswell, and B. Liu, editors, *WSDM*, pages 81–90. ACM, 2010.

[13] M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola. Cofirank, maximum margin matrix factorization for collaborative ranking. In *Advances in Neural Information Processing Systems*, 2007.

[14] M. Weimer, A. Karatzoglou, and A. Smola. Improving maximum margin matrix factorization. *Machine Learning*, 72(3):263–276, 2008.