Chapter 6
Item Recommendation

Recommending content is an important task in many information systems. For example online shopping websites like Amazon give each customer personalized recommendations of products that the user might be interested in. Other examples are video portals like YouTube that recommend videos to visitors. Personalization is attractive both for content providers, who can increase sales or views, and for customers, who can find interesting content more easily. In this chapter, we focus on item recommendation where the task is to create a user-specific ranking for a set of items. Preferences of users about items are learned from the user's past interaction with the system – e.g. his buying history, viewing history, etc. Thus, the context in item recommenders is the user and user-aware rankings should be generated.

Recommender systems are an active topic of research. Most recent work is on scenarios where users provide explicit feedback, e.g. in terms of ratings. Nevertheless, in real-world scenarios most feedback is not explicit but implicit. Implicit feedback is tracked automatically, like monitoring clicks, view times, purchases, etc. Thus it is much easier to collect, because the user has not to express his taste explicitly. In fact implicit feedback is already available in almost any information system – e.g. web servers record any page access in log files.

We start with introducing the related work in the field of item recommendation. Then, we discuss the problem setting and show how it fits into the context-aware ranking framework. Afterwards, we apply the *Bayesian Context-aware Ranking* (BCR) framework to personalization, thus we get *Bayesian Personalized Ranking* (BPR). We show the optimization criterion and learning algorithm for this instance of BCR. Then, we describe how the two popular item recommendation models k-nearest neighbor and matrix factorization can be learned with BPR. In the evaluation, we compare the quality of these approaches to the state-of-the-art recommenders of weighted regularized matrix factorization and cosine kNN.

6.1 Related Work

The most popular model for recommender systems is k-nearest neighbor (kNN) collaborative filtering (Deshpande and Karypis, 2004). Traditionally, the similarity

S. Rendle: Context-Aware Ranking with Factorization Models, SCI 330, pp. 69–84.
springerlink.com

matrix of kNN is computed by heuristics – e.g. the Pearson correlation – but in recent work (Koren, 2008) the similarity matrix is treated as model parameters and is learned specifically for the task. Recently, matrix factorization (MF) has become very popular in recommender systems both for implicit and explicit feedback. In early work (Sarwar et al, 2002) singular value decomposition (SVD) has been proposed to learn the feature matrices. MF models learned by SVD have shown to be very prone to overfitting (Kurucz et al, 2007). Thus regularized learning methods have been proposed. For item prediction Hu et al (2008) and Pan et al (2008) propose a regularized least-square optimization with case weights (WR-MF). The case weights can be used to reduce the impact of negative examples. Hofmann (2004) proposes a probabilistic latent semantic model for item recommendation. Schmidt-Thieme (2005) converts the problem into a multi-class problem and solves it with a set of binary classifiers.

Even though all the work on item prediction discussed above is evaluated on personalized ranking datasets, none of these methods directly optimizes its model parameters for ranking. Instead they optimize to predict if an item is selected by a user or not. In our work we derive an optimization criterion for personalized ranking that is based on pairs of items (i.e. the user-specific order of two items). We will show how state-of-the-art models like MF or adaptive kNN can be optimized with respect to this criterion to provide better ranking quality than with usual learning methods. A detailed discussion of the relationship between our approach and the WR-MF approach of Hu et al (2008) and Pan et al (2008) as well as maximum margin matrix factorization (Srebro et al, 2005; Weimer et al, 2008) can be found in section 6.5.

In this paper, we focus on offline learning of the model parameters. Extending the learning method to online learning scenarios – e.g. a new user is added and his history increases from 0 to 1, 2, ... feedback events – has already been studied for MF for the related task of rating prediction Rendle and Schmidt-Thieme (2008). The same fold-in strategy can be used for BPR.

There is also related work on learning to rank with non-collaborative models. One direction is to model distributions on permutations (Kondor et al, 2007; Huang et al, 2008). Burges et al (2005) optimize a neural network model for ranking using gradient descent. All these approaches learn only one ranking – i.e. they are non-personalized. In contrast to this, our models are collaborative models that learn personalized rankings, i.e. one individual ranking per user. In our evaluation, we show empirically that in typical recommender settings our personalized BPR model outperforms even the theoretical upper bound for non-personalized ranking.

6.2 Personalized Ranking from Implicit Feedback

The task of personalized ranking is to provide a user with a ranked list of items. This is also called item recommendation. An example is an online shop that wants to recommend a personalized ranked list of items that the user might want to buy.

In this work, we investigate scenarios where the ranking has to be inferred from the implicit behavior (e.g. purchases in the past) of the user. Interesting about implicit feedback systems is that only positive observations are available. The non-observed user-item pairs – e.g. a user has not bought an item yet – are a mixture of real negative feedback (the user is not interested in buying the item) and missing values (the user might want to buy the item in the future).

6.2.1 Formalization

For easier readability, we adapt the formalization of chapter 3 to the special case of item recommendation, e.g. we use the more meaningful symbols U for users and I for items instead of X_1 and X_2.

Item recommendation is a two mode problem ($m = 2$). Let $U = X_1$ be the set of all users and $I = X_2$ the set of all items. In our scenario, the observed training data is binary set data – i.e. it can be written as $S \subseteq U \times I$ (see the topmost matrix in figure 6.1). Examples for such feedback are purchases in an online shop, views in a video portal or clicks on a website. The task of the recommender system is now to provide the user with a personalized ranking $\succ_u \subset I^2$ of all items.

6.2.2 Analysis of the Problem Setting

As we have indicated before, in implicit feedback systems only positive classes are observed. The remaining data is a mixture of actually negative and missing values. The most common approach for dealing with the missing value problem is to ignore all of them but then typical machine learning models are unable to learn anything, because they cannot distinguish between the two levels anymore. Machine learning approaches for item recommenders (Hu et al, 2008; Pan et al, 2008) typically create the training data from S by giving all pairs $(u,i) \in S$ a positive class label and all other combinations in $(U \times I) \setminus S$ a negative one – that means they try to reconstruct the observed data S of the past. As we have discussed in chapter 3, the problem with this approach is that all elements the model should rank in the future ($(U \times I) \setminus S$) are presented to the learning algorithm as negative feedback during training. That means a model with enough expressiveness (that can fit the training data exactly) cannot rank at all as it predicts only negative class values. The only reason why such machine learning methods can predict rankings are strategies to prevent overfitting, like regularization.

Instead, our approach is to see the task as a ranking problem rather than a classification problem and to create pairwise training data D_S (see chapter 3). This process is illustrated in figure 6.1. If an item i has been viewed by user u – i.e. $(u,i) \in S$ – then we assume that the user prefers this item over all other non-observed items. E.g. in Figure 6.1 user u_1 has viewed item i_2 but not item i_1, so we assume that this user prefers item i_2 over i_1: $i_2 \succ_{u_1} i_1$. For items that have both been seen by a user,

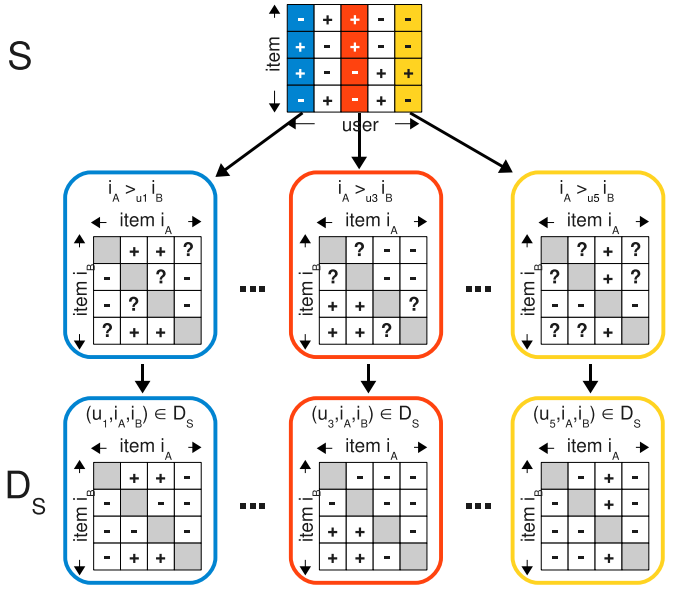

Fig. 6.1 Item Recommendation: the observed data S is binary (*top*). For each user, we try to find indication in the data S if the user prefers one item over the other (*middle*). The inferred training data D_S is then shown at the bottom. In all figures, a '+' means that the element is in the set, a '-' means it is not in the set and a '?' means that it is unknown whether or not it is in the set.

we cannot infer any preference. The same is true for two items that a user has not seen yet (e.g. item i_1 and i_4 for user u_1). Now, D_S can be created according to eq. (3.21), which reads the following for item recommendation:

$$D_S := \{(u,i,j) : (u,i) \in S \wedge (u,j) \notin S\} \tag{6.1}$$

Note that even though D_S explicitly contains only positive pairs, it also contains implicitly the negative ones because pairs are antisymmetric – but it does not contain the pairs that are neither positive nor negative (missing values), see figure 6.1.

In total, our approach has two advantages:

1. The pairwise training data consists of both positive and negative pairs and missing values. The missing values between two non-observed items are exactly the item pairs that have to be ranked in the future. That means from a pairwise point of view the training data D_S and the test data is disjoint. This does not hold for the classification approach that tries to reconstruct S and $(U \times I) \setminus S$.
2. The training data is created for the actual objective of ranking, i.e. the observed subset D_S of $\succ_u$ is used as training data.

6.3 Learning Personalized Ranking

In this section, we transfer the general BCR method to item recommendation. As we have stated before, for item recommendation, the context is the user, i.e. $\mathscr{C} = U$. Thus, we refer to Bayesian Context-aware Ranking (BCR) as *Bayesian Personalized Ranking* (BPR). We will both present the optimization criterion BPR-OPT and the learning algorithm BPR-LEARN.

6.3.1 Optimization Criterion (BPR-OPT)

Analogous to BCR (see section 4.1), we can derive BPR-OPT which is the MAP estimator for the model parameters Θ of an estimator $\hat{y}$:

$$\underset{\Theta}{\operatorname{argmax}}\, p(\Theta \mid \succ) = \underset{\Theta}{\operatorname{argmax}}\, p(\succ \mid \Theta)\, p(\Theta) \tag{6.2}$$

where the probability for $\succ$ is:

$$p(\succ \mid \Theta) = \prod_{(u,i,j) \in U \times I^2} p(i \succ_u j \mid \Theta)^{2\,\delta(i \succ_u j)} \tag{6.3}$$

Again, we assume that $\hat{y}$ is used as an estimator for $\succ$ (see section 4.1.3) and thus:

$$p(i \succ_u j \mid \Theta) := \sigma(\hat{y}(u,i) - \hat{y}(u,j)) \tag{6.4}$$

And for the prior, we assume a normal distribution, thus:

$$p(\Theta) = \prod_{\theta \in \Theta} \sqrt{\frac{\lambda_\theta}{\pi}} \exp\left(-\lambda_\theta\, \theta^2\right) \tag{6.5}$$

Putting everything together, leads to BPR-OPT which is defined as the MAP estimator given the training data:

$$\underset{\Theta}{\operatorname{argmax}}\, \text{BPR-OPT} := \underset{\Theta}{\operatorname{argmax}} \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{y}(u,i) - \hat{y}(u,j)) - \sum_{\theta \in \Theta} \frac{1}{2} \lambda_\theta\, \theta^2 \tag{6.6}$$

where λ_θ is a model specific regularization parameter.

6.3.2 Learning Algorithm (BPR-LEARN)

Next, we specialize BCR-LEARN to item recommendation to derive BPR-LEARN. This algorithm is based on stochastic gradient descent where cases $(u,i,j) \in D_S$ are drawn by bootstrapping. The benefits of a stochastic approach with bootstrapping over the standard approach of computing the full gradient or user-wise stochastic descent are discussed in section 4.2.1.

Gradients

For performing gradient descent, the partial derivative of BPR-OPT with respect to the model parameters θ given a case $(u,i,j) \in D_S$ is:

$$\frac{\partial}{\partial \theta} \text{BPR-OPT} = \delta_{u,i,j} \frac{\partial}{\partial \theta} (\hat{y}(u,i) - \hat{y}(u,j)) - \lambda_\theta \theta \tag{6.7}$$

with:

$$\delta_{u,i,j} := (1 - \sigma(\hat{y}(u,i) - \hat{y}(u,j))) \tag{6.8}$$

Drawing of Cases

It is not feasible to explicitly enumerate all triples in D_S. Instead, we can draw cases (u,i,j) from D_S without enumerating them. This can be done by first drawing a tuple (u,i) from S and then drawing a negative item j. The drawing of a negative item can be performed by (1) drawing j from I and (2) rejecting it if $(u,j) \in S$. Rejection is very unlikely because most items are not purchased/ bought/ etc. by the user. More details about this can be found in section 4.2.3.

BPR-Learn

Algorithm 2 shows the learning method BPR-LEARN for optimizing the model parameters Θ with respect to BPR-OPT. This algorithm is an adaption of BCR-LEARN (see algorithm 1) to the context of personalization. After initializing the model parameters Θ, the parameters are learned iteratively. First a case is drawn as described before and then gradient descent is performed on the related parameters. For termination of the main-loop, we searched the best number of iterations on a holdout set. This corresponds to early stopping and can be seen as an additional way of regularizing.

6.4 Item Recommendation Models

In the following, we describe two state-of-the-art model classes for item recommendation and show how they can be learned with our proposed BPR methods. We have chosen the two diverse model classes of matrix factorization (Hu et al, 2008; Rennie and Srebro, 2005) and learned k-nearest-neighbor (Koren, 2008). Both classes try to model the hidden preferences of a user on an item. Their prediction is a real number $\hat{y}(u,i)$ per user-item-pair (u,i). As both U and I are categorical domains, one can express $y : U \times I \to \mathbb{R}$ as a matrix $Y \in \mathbb{R}^{|U| \times |I|}$. Thus, we will use this notation in the following and write $y_{u,i} := y(u,i)$.

It is important to note that even though we use the same models as in other work, we optimize them against another criterion. This will lead to a better ranking

Algorithm 2 BPR-LEARN

Input: training data S, learning rate α, regularization parameters λ_θ
Output: model parameters Θ

initialize Θ from $\mathcal{N}(0, \sigma^2)$
repeat
  draw (u,i) uniformly from S
  draw j from $\{l : (u,l) \notin S\}$
  $\delta_{u,i,j} \leftarrow (1 - \sigma(\hat{y}(u,i) - \hat{y}(u,j)))$
  for $\theta \in \Theta$ **do**
    $\theta \leftarrow \theta + \alpha \left(\delta_{u,i,j} \frac{\partial}{\partial \theta} (\hat{y}(u,i) - \hat{y}(u,j)) - \lambda_\theta \, \theta \right)$
  end for
until convergence
return Θ

because our criterion is optimal for the ranking task. It does not try to regress a single predictor $\hat{y}_{u,i}$ to a single number but instead tries to classify the ranking of two predictions, i.e. $\hat{y}_{u,i} > \hat{y}_{u,j}$.

6.4.1 Matrix Factorization

In chapter 5, we have presented three tensor factorization models and have shown that for two mode tensors, PARAFAC and PITF corresponds to matrix factorization. In this section, we show how the matrix factorization model can be used for item recommendation.

The problem of predicting $\hat{y}_{u,i}$ can be seen as the task of estimating a real valued matrix $Y \in \mathbb{R}^{|U| \times |I|}$. With matrix factorization the target matrix Y is approximated by the matrix product of two low-rank matrices $W \in \mathbb{R}^{|U| \times k}$ and $H \in \mathbb{R}^{|I| \times k}$:

$$\hat{Y} := W \cdot H^t$$

where k is the dimensionality/rank of the approximation. Each vector/ row $\mathbf{w}_u$ in W can be seen as a feature vector describing a user u and similarly each row $\mathbf{h}_i$ of H describes an item i. Thus the prediction formula can also be written as:

$$\hat{y}_{u,i} = \langle \mathbf{w}_u, \mathbf{h}_i \rangle = \sum_{f=1}^{k} w_{u,f} \cdot h_{i,f}$$

Besides the dot product $\langle \cdot, \cdot \rangle$ in general any kernel can be used like in (Rendle and Schmidt-Thieme, 2008). The model parameters for matrix factorization are $\Theta = (W, H)$.

In general the best approximation of $\hat{Y}$ to Y with respect to dense element-wise least-square is achieved by the singular value decomposition (SVD). For machine learning tasks, it is known that SVD overfits (Kurucz et al, 2007) and therefore many other matrix factorization methods have been proposed, including regularized least square optimization, non-negative factorization, maximum margin factorization, etc.

For the task of ranking, i.e. estimating whether a user prefers one item over another, a better approach is to optimize against the BPR-OPT criterion. This can be achieved by using our proposed algorithm BPR-LEARN. As stated before for optimizing with BPR-LEARN, only the gradient of $\hat{y}_{u,i}$ with respect to every model parameter θ has to be known. For the matrix factorization model the derivatives given a case $(u,i,j) \in D_S$ are:

$$\frac{\partial}{\partial \theta}(\hat{y}_{u,i} - \hat{y}_{u,j}) = \begin{cases} (h_{i,f} - h_{j,f}) & \text{if } \theta \text{ is } w_{u,f}, \\ w_{u,f} & \text{if } \theta \text{ is } h_{i,f}, \\ -w_{u,f} & \text{if } \theta \text{ is } h_{j,f}, \\ 0 & \text{else} \end{cases} \tag{6.9}$$

Furthermore, we use three regularization constants: one λ_W for the user features W; for the item features H we have two regularization constants, λ_{H^+} that is used for positive updates on $h_{i,f}$, and λ_{H^-} for negative updates on $h_{j,f}$.

6.4.2 Adaptive k-Nearest-Neighbor

In addition to the factorization models on which we are focused in this book, we describe here another modelling approach that can be applied for two-mode problems.

Nearest-neighbor methods are very popular in collaborative filtering. They rely on a similarity measure between either items (item-based) or users (user-based). In the following we describe item-based methods as they usually provide better results, but user-based methods work analogously. The idea is that the prediction for a user u and an item i depends on the similarity of i to all other items the user has seen in the past – i.e. I_u:

$$I_u := \{i : (u,i) \in S\} \tag{6.10}$$

Often only the k most similar items of I_u are regarded – the *k-nearest neighbors*. If the similarities between items are chosen carefully, one can also compare to all items in I_u. For item prediction the model of item-based k-nearest-neighbor is:

$$\hat{y}_{u,i} = \sum_{l \in I_u \wedge l \neq i} c_{i,l} \tag{6.11}$$

where $C : I \times I$ is the symmetric item-correlation/ item-similarity matrix. Hence the model parameters of kNN are $\Theta = C$.

The common approach for choosing C is by applying a heuristic similarity measure, e.g. cosine vector similarity:

$$c_{i,j}^{\text{cosine}} := \frac{|U_i \cap U_j|}{\sqrt{|U_i| \cdot |U_j|}} \tag{6.12}$$

with:

$$U_l := \{u : (u,l) \in S\} \tag{6.13}$$

A better strategy is to adapt the similarity measure C to the problem by learning it. This can be either done by using C directly as model parameters or if the number of items is too large, one can learn a factorization HH^t of C with $H \in \mathbb{R}^{|I| \times k}$. In the following and also in our evaluation we use the first approach of learning C directly without factorizing it.

Again for optimizing the kNN model for ranking, we apply the BPR optimization criterion and use the BPR-LEARN algorithm. For applying the algorithm, the gradient of $\hat{y}_{u,i} - \hat{y}_{u,j}$ with respect to the model parameters C is:

$$\frac{\partial}{\partial \theta}(\hat{y}_{u,i} - \hat{y}_{u,j}) = \begin{cases} +1 & \text{if } \theta \in \{c_{i,l}, c_{l,i}\} \wedge l \in I_u \wedge l \neq i, \\ -1 & \text{if } \theta \in \{c_{j,l}, c_{l,j}\} \wedge l \in I_u \wedge l \neq j, \\ 0 & \text{else} \end{cases}$$

We have two regularization constants, λ_+ for updates on $c_{i,l}$, and λ_- for updates on $c_{j,l}$.

6.5 Relations to Other Methods

We discuss the relations of our proposed methods for ranking to two further item recommendation models. The first one is an element-wise weighted least-square error (eq. (4.25)) and the second one uses pairwise hinge loss (eq. (4.17)).

6.5.1 Weighted Regularized Matrix Factorization (WR-MF)

Both Pan et al (2008) and Hu et al (2008) have presented a matrix factorization method for item prediction from implicit feedback. Thus the model class is the same as we have described in section 6.4.1, i.e. $\hat{Y} := WH^t$ with the matrices $W \in \mathbb{R}^{|U| \times k}$ and $H \in \mathbb{R}^{|U| \times k}$. The optimization criterion and learning method differ substantially from our approach. Their method is an adaption of an SVD, which minimizes the square-loss. Their extensions are regularization to prevent overfitting and weights in the error function to increase the impact of positive feedback. In total their optimization criterion is:

$$\sum_{u \in U} \sum_{i \in I} c_{u,i} \left(\langle \mathbf{w}_u, \mathbf{h}_i \rangle - \delta((u,i) \in S) \right)^2 + \lambda ||W||_f^2 + \lambda ||H||_f^2 \tag{6.14}$$

where $c_{u,i}$ are not model parameters but apriori given weights for each tuple (u,i). Hu et al. have additional data to estimate $c_{u,i}$ for positive feedback and they set $c_{u,i} = 1$ for the rest. Pan et al. suggest to set $c_{u,i} = 1$ for positive feedback and choose lower constants for the rest.

First of all, it is obvious that this optimization is on instance level (one item) instead of pair level (two items) as BPR. Apart from this, their optimization is a least-square which is known to correspond to the MLE for normally distributed random

variables. However, the task of item recommendation is actually not a regression (quantitative), but a classification (qualitative) one, so the logistic optimization is more appropriate.

A strong point of WR-MF is that it can be learned in $O(\text{iter}(|S|k^2 + k^3(|I| + |U|)))$ provided that $c_{u,i}$ is constant for non-positive pairs. Our evaluation indicates that BPR-LEARN usually converges after a subsample of $|D_S|$ single update steps even though there are much more triples to learn from.

6.5.2 Maximum Margin Matrix Factorization for Ordinal Ranking

Weimer et al (2008) use the maximum margin matrix factorization method (MMMF) (Srebro et al, 2005) for ordinal ranking. Their MMMF is designed for scenarios with explicit feedback in terms of ratings. Even though their ranking MMMF is not intended for implicit feedback datasets, one could apply it in our scenario by giving all non-observed items the 'rating' 0 and the observed ones a 1. With these modifications their optimization criterion to be minimized would be quite similar to BPR applied for matrix factorization:

$$\sum_{(u,i,j)\in D_S} \max(0, 1 - \langle \mathbf{w}_u, \mathbf{h}_i - \mathbf{h}_j \rangle) + \lambda_w ||W||_f^2 + \lambda_h ||H||_f^2 \tag{6.15}$$

One difference is that the error functions differ – our hinge loss is smooth and motivated by the MLE. Additionally, our BPR-OPT criterion and BPR-LEARN algorithm is generic and can be applied to several models, whereas their method is specific for MF.

Besides this, their learning method for MMMF differs from our generic approach BPR-LEARN. Their learning method is designed to work with sparse explicit data, i.e. they assume that there are many missing values and thus they assume to have much less pairs than in an implicit setting. But when their learning method is applied to implicit feedback datasets, the data has to be densified like described above and the number of training pairs D_S is in $O(|S||I|)$ (see section 3.3.2). Our method BPR-LEARN can handle this situation by bootstrapping from D_S.

6.6 Evaluation

In our evaluation, we compare learning with BPR to other learning approaches. We have chosen the two popular model classes of matrix factorization (MF) and k-nearest-neighbor (kNN). Regularized MF models are known to outperform (Rennie and Srebro, 2005) many other models including the Bayesian models URP (Marlin, 2004) and PLSA (Hofmann, 2004) for the related task of collaborative rating prediction. In our evaluation, the matrix factorization models are learned by three different methods, i.e. SVD-MF, WR-MF (Hu et al, 2008; Pan et al, 2008) and our BPR-MF. For kNN (see eq. (6.11)), we compare cosine vector similarity

(Cosine-kNN) to a model that has been optimized using our BPR method (BPR-kNN)[1]. Additionally, we report results for the baseline most-popular, that weights each item user-independently, e.g.: $\hat{y}_{u,i}^{\text{most-pop}} := |U_i|$. Furthermore, we give the theoretical upper bound on the AUC for any non-personalized ranking method ($\text{np}_{\max}$).

6.6.1 Datasets

We use two datasets of two different applications. The *Rossmann* dataset is from an online drug store[2]. It contains the buying history of 10,000 users on 4000 items. In total 426,612 purchases are recorded. The task is to predict a personalized list of the items the user wants to buy next. The second dataset is the DVD rental dataset of *Netflix*[3]. This dataset contains the rating behavior of users, where a user provides explicit ratings 1 to 5 stars for some movies. As we want to solve an implicit feedback task, we removed the rating scores from the dataset. Now the task is to predict which movies a user rates. Again we are interested in a personalized ranked list starting with the movie that is most likely to be rated. For Netflix we have created a subsample of 10,000 users, 5000 items containing 565,738 rating actions. We draw the subsample such that every user has at least 10 items ($\forall u \in U : |I_u| \geq 10$) and each item has at least 10 users ($\forall i \in I : |U_i| \geq 10$).

6.6.2 Evaluation Methodology

We use the leave-one-out evaluation scheme, where we remove for each user randomly one action (one user-item pair) from his history, i.e. we remove one entry from I_u per user u. This results in a disjoint train set S_{train} and test set S_{test}. The models are then learned on S_{train} and their predicted personalized ranking is evaluated on the test set S_{test} by the average AUC statistic (see section 3.5). A higher value of the AUC indicates a better quality. The trivial AUC of a random guess method is 0.5 and the best achievable quality is 1.

We repeated all experiments 10 times by drawing new train/test splits in each round. The hyperparameters for all methods are optimized via grid search on the train/test split of the first round and afterwards the hyperparameters are kept constant in the remaining 9 repetitions.

On Netflix, the training runtime of the largest factorization models (k=128) was 174 minutes for BPR-MF (134 on Rossmann) and 117 minutes for WR-MF (97 on Rossmann). For BPR-kNN, the runtime was 68 minutes on Netflix and 139 on Rossmann.

[1] One might also apply the weighted regularized least square scheme of WR-MF to kNN and obtain 'WR-kNN'. To the best of our knowledge, this has not been done yet and there is no proposal for a feasible learning algorithm for WR-kNN like there exists for WR-MF. Without such a speedup, the WR-kNN iterates over the full matrix with $|U||I|$ entries.

[2] `http://www.rossmannversand.de/`

[3] `http://www.netflix.com/`

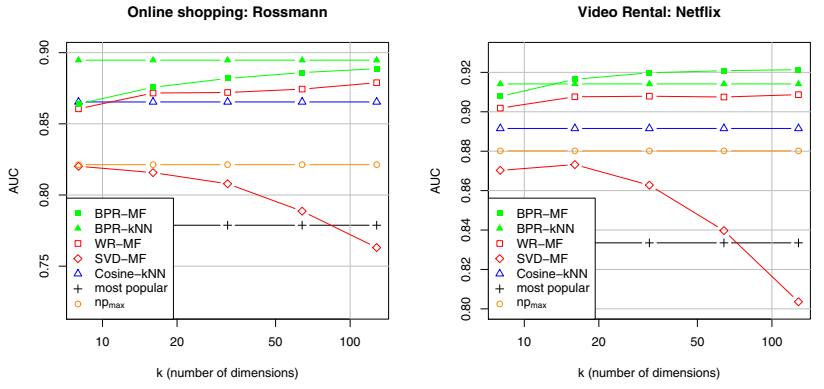

Fig. 6.2 Area under the ROC curve (AUC) prediction quality for the Rossmann dataset and a Netflix subsample. Our BPR optimization for matrix factorization BPR-MF and k-nearest neighbor BPR-kNN are compared against weighted regularized matrix factorization (WR-MF) (Hu et al, 2008; Pan et al, 2008), singular value decomposition (SVD-MF), k-nearest neighbor (Cosine-kNN) (Deshpande and Karypis, 2004) and the most-popular model. For the factorization methods BPR-MF, WR-MF and SVD-MF, the model dimensions are increased from 8 to 128 dimensions. Finally, np_{max} is the theoretical upper bound for any non-personalized ranking method.

6.6.3 Results and Discussion

Figure 6.2 shows the AUC quality of all models on the two datasets. First of all, you can see that the two BPR optimized methods outperform all other methods in prediction quality. Comparing the same models among each other one can see the importance of the optimization method. For example all MF methods (SVD-MF, WR-MF and BPR-MF) share exactly the same model, but their prediction quality differs a lot. Even though SVD-MF is known to yield the best fit on the training data with respect to element-wise least square, it is a poor prediction method for machine learning tasks as it overfits the training data. This can be seen as the quality of SVD-MF decreases with an increasing number of dimensions (expressiveness). WR-MF is a more successful learning method for the task of ranking. Due to regularization its performance does not drop but steadily rises with an increasing number of dimensions. But BPR-MF outperforms WR-MF clearly for the task of ranking on both datasets. For example on Netflix a MF model with 8 dimensions optimized by BPR-MF achieves comparable quality as a MF model with 128 dimensions optimized by WR-MF.

To summarize, our results show the importance of optimizing model parameters to the right criterion. The empirical results indicate that our BPR-OPT criterion learned by BPR-LEARN outperforms the other state-of-the-art methods for personalized ranking from implicit feedback. The results are justified by the analysis of the problem (see section 6.2.2) and by the theoretical derivation of BPR-OPT from the MLE over pairs.

6.6.4 Non-personalized Ranking

Next, we compare the AUC quality of our personalized ranking methods to the best possible non-personalized ranking method. In contrast to our personalized ranking methods, a non-personalized ranking method creates the same ranking $\succ$ for all users. We compute the theoretical upper-bound $\text{np}_{\max}$ for any non-personalized ranking method by optimizing the ranking $\succ$ on the test set S_{test}. Figure 6.2 shows that even simple personalized methods like Cosine-kNN outperform the upper-bound $\text{np}_{\max}$ — and thus also all non-personalized methods — largely.

Computation of the AUC of $\text{np}_{\max}$

Computing a global (non-personalized) ranking $\hat{y}$ that is AUC-optimal on test is not trivial because within a user the train examples are not allowed to be rerecommended and different users have different training sizes. This leads to a slightly different weighting in the AUC computation of pair comparisons for each user. So instead of searching an optimal $\hat{y}^*$, we computed an upper-bound but non-tight estimate on the AUC score of $\text{np}_{\max}$.

First, we define the evaluation sets of items for each user and the set of all evaluated users:

$$I_u^+ := \{i \in I : (u,i) \in S_{\text{test}} \wedge (u,i) \notin S_{\text{train}}\} \tag{6.16}$$

$$I_u^- := \{i \in I : (u,i) \notin S_{\text{test}} \wedge (u,i) \notin S_{\text{train}}\} \tag{6.17}$$

$$U^+ := \{u \in U : I_u^+ \neq \emptyset\} \tag{6.18}$$

With this, the AUC quality measure can be written as:

$$\text{AUC}(\hat{y}) = \frac{1}{|U^+|} \sum_{u \in U^+} \frac{1}{|I_u^+||I_u^-|} \sum_{i \in I_u^+} \sum_{j \in I_u^-} \delta(\hat{y}(u,i) > \hat{y}(u,j)) \tag{6.19}$$

The best non-personalized ranking on test $\hat{y}^*$ depends not on the user and has the following AUC:

$$\text{AUC}(\hat{y}^*) = \frac{1}{|U^+|} \sum_{u \in U^+} \frac{1}{|I_u^+||I_u^-|} \sum_{i \in I_u^+} \sum_{j \in I_u^-} \delta(\hat{y}^*(i) > \hat{y}^*(j)) \tag{6.20}$$

Finding the best order is non-trivial. Instead, we create an upper-bound by evaluating the contribution of a decision about a pair $(i,j) \in I^2$ on the AUC. For each pair (i,j) either i is ranked in front of j or vice versa. If the estimated ranking is $i \succ j$ then the contribution on the AUC is:

$$a_{i,j} = \frac{1}{|U^+|} \sum_{u \in U^+} \frac{1}{|I_u^+||I_u^-|} \delta(i \in I_u^+ \wedge j \in I_u^-) \tag{6.21}$$

The inverse ranking would have the contribution:

$$a_{j,i} = \frac{1}{|U^+|} \sum_{u \in U^+} \frac{1}{|I_u^+||I_u^-|} \delta(j \in I_u^+ \wedge i \in I_u^-) \tag{6.22}$$

As the ranking $\text{np}_{\max}$ has to decide if i is preferred over j, only one of the two scores can be used. This means, an upper bound estimate on the AUC is to use for each comparison the larger value:

$$AUC^{\text{upper bound}} = \frac{1}{2} \sum_{i \in I} \sum_{j \in I, i \neq j} \max\left(a_{i,j}, a_{j,i}\right) \tag{6.23}$$

Note that this is a non-tight upper bound on the AUC as the pairwise decisions might not be total (e.g. not transitive) and so this does not have to lead to a total order (ranking).

In our experiments, this non-tight upper bound is close to the true AUC value of $\text{np}_{\max}$. This can be seen by calculating the interval where $\text{np}_{\max}$ is guaranteed to lie in. $AUC^{\text{upper bound}}$ gives an upper bound; a lower bound can be computed by ranking the items by how often they appear in the test set ('most-popular on test'). In our experiments both AUC scores are quite similar, e.g. on Netflix with most-popular on test 0.8794 vs. $AUC^{\text{upper bound}} = 0.8801$.

6.6.5 Practical Impact

Finally, we discuss the practical impact of the improvements. Linden et al (2003) found that in the Amazon online shop, the conversion rates[4] generated by a cosine-knn recommender *vastly exceed* the ones of the most-popular recommender (*top-seller lists*). Relating this finding with the ranking accuracy in our lab experiments (e.g. on the Netflix dataset), it would mean that the difference of cosine-knn (11% error) to most-popular (17% error) is likely to have a huge practical impact, so we might conclude that a further decrease from 11% error (cosine-knn) to 8% error (BPR-MF) has also a significant impact.

6.7 Conclusion

In this chapter, we have presented how to apply the context-aware ranking method to item recommendation. We have shown that item recommendation is a context-aware ranking problem from incomplete data. This allows to transfer the BCR-OPT and BCR-LEARN methods to personalized ranking (BPR). We have demonstrated how this generic method can be applied to the two state-of-the-art item recommender models of matrix factorization and adaptive kNN. In our evaluation, we have shown

[4] The convertion rate is one of the most important measures to assess the practical success of a recommender system.

empirically that for the task of personalized ranking, models learned by BPR outperform the same models that are optimized with respect to other criteria. Our results show that the prediction quality does not only depend on the model but also largely on the optimization criterion. Both our theoretical and empirical results indicate that the BPR optimization method is the right choice for the important task of item recommendation.

References

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: ICML 2005: Proceedings of the 22nd International Conference on Machine Learning, pp. 89–96. ACM Press, New York (2005)

Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. In: ACM Transactions on Information Systems, vol. 22(1), Springer-Verlag, Heidelberg (2004)

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

Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: IEEE International Conference on Data Mining (ICDM 2008), pp 263–272 (2008)

Huang, J., Guestrin, C., Guibas, L.: Efficient inference for distributions on permutations. In: Platt, J., Koller, D., Singer, Y., Roweis, S. (eds.) Advances in Neural Information Processing Systems, vol. 20, pp. 697–704. MIT Press, Cambridge (2008)

Kondor, R., Howard, A., Jebara, T.: Multi-object tracking with representations of the symmetric group. In: Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics, San Juan, Puerto Rico (2007)

Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: KDD 2008: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 426–434. ACM, New York (2008)

Kurucz, M., Benczúr, A.A., Torma, B.: Methods for large scale svd with missing values. In: KDDCup 2007 (2007)

Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. Internet Computing 7(1), 76–80 (2003)

Marlin, B.: Modeling user rating profiles for collaborative filtering. In: Thrun, S., Saul, L., Schölkopf, B. (eds.) Advances in Neural Information Processing Systems, vol. 16, MIT Press, Cambridge (2004)

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

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

Rennie, J.D.M., Srebro, N.: Fast maximum margin matrix factorization for collaborative prediction. In: ICML 2005: Proceedings of the 22nd International Conference on Machine Learning, pp. 713–719. ACM, New York (2005)

Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Incremental singular value decomposition algorithms for highly scalable recommender systems. In: Proceedings of the 5th International Conference in Computers and Information Technology (2002)

Schmidt-Thieme, L.: Compound classification models for recommender systems. In: IEEE International Conference on Data Mining (ICDM 2005), pp. 378–385 (2005)

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

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