A Boosting Algorithm for Item Recommendation with Implicit Feedback

Yong Liu[1,2], **Peilin Zhao**[3], **Aixin Sun**[2], **Chunyan Miao**[1,2]
[1]Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY),
Nanyang Technological University, Singapore
[2]School of Computer Engineering, Nanyang Technological University, Singapore
[3]Institute for Infocomm Research, A*STAR, Singapore
{liuy0054@e., axsun@, ascymiao@}ntu.edu.sg, zhaop@i2r.a-star.edu.sg

Abstract

Many recommendation tasks are formulated as top-N item recommendation problems based on users' implicit feedback instead of explicit feedback. Here explicit feedback refers to users' ratings to items while implicit feedback is derived from users' interactions with items, *e.g.,* number of times a user plays a song. In this paper, we propose a boosting algorithm named AdaBPR (Adaptive Boosting Personalized Ranking) for top-N item recommendation using users' implicit feedback. In the proposed framework, multiple homogeneous component recommenders are linearly combined to create an ensemble model, for better recommendation accuracy. The component recommenders are constructed based on a fixed collaborative filtering algorithm by using a re-weighting strategy, which assigns a dynamic weight distribution on the observed user-item interactions. AdaBPR demonstrates its effectiveness on three datasets compared with strong baseline algorithms.

1 Introduction

In recent years, recommender systems have been widely adopted to help users discover most useful information from a massive amount of data. Lots of algorithms have been proposed and many of them are used to power recommender systems in various application domains [Su and Khoshgoftaar, 2009]. Generally speaking, there are two main categories of recommendation tasks: *rating prediction* and *item recommendation*. The objective of rating prediction is to predict the rating that a user may give to an item that she has not interacted with before. Examples include movie rating prediction in Netflix [Bell and Koren, 2007] and business rating prediction in Yelp [Hu *et al.*, 2014]. For item recommendation, recommender system recommends a user a list of items that she might prefer. Product recommendation in Amazon [Linden *et al.*, 2003], friend recommendation in online social networks [Chen *et al.*, 2009], and location recommendation in Foursquare [Liu *et al.*, 2014] are examples of item recommendation tasks.

For item recommendation with implicit feedback, users' preferences on items are unobservable. For example, in the music recommendation task[1], the number of times a user plays a song is observable (*i.e.,* implicit feedback) but not her preference rating to the song (*i.e.,* explicit feedback). Recommendation models are then built based on some pre-defined preference assumptions, *e.g.,* the more number of times a user plays a song, the more the user likes the song. Such assumptions may not accurately describe users' preferences. Moreover, the observed user-item interaction data is generally very sparse, which makes the preference modeling even more challenging. As the result, existing solutions often deliver unsatisfactory recommendation accuracies.

Instead of directly addressing these challenges in item recommendation with implicit feedback, we consider an alternative approach in this paper. We exploit boosting technique to improve the recommendation accuracy by combining the recommendation power of multiple "weak recommenders". Boosting technique was originally proposed to improve the accuracy of classification models by combining multiple weak classifiers [Freund and Schapire, 1995]. It was then extended to solve ranking tasks. For example, AdaRank is a boosting framework that optimizes ranking metrics such as the Area Under the ROC Curve (AUC), the Mean Average Precision (MAP), and the Normalized Discounted Cumulative Gain (NDCG) [Xu and Li, 2007]. Recent studies show that accuracies of previous recommendation methods can be improved by utilizing boosting techniques [Jiang *et al.*, 2013; Cheng *et al.*, 2014; Wang *et al.*, 2014]. However, all existing solutions are based on the recommendation models designed for rating prediction tasks, which are not optimal for item recommendation with implicit feedback.

In this paper, we propose a boosting framework, namely Adaptive Boosting Personalized Ranking (AdaBPR), for item recommendation with users' implicit feedback. In this framework, multiple homogeneous component recommenders are linearly combined to achieve more accurate recommendation. The component recommenders are learned based on a re-weighting strategy that assigns a dynamic weight to each observed user-item interaction. We evaluated AdaBPR and four baseline methods on three datasets, namely MovieLens-100K, MovieLens-1M, and the Taste Profile Subset of the Million Song Dataset. AdaBPR achieves the best recommendation accuracy on all the three datasets, evaluated on three

[1]http://labrosa.ee.columbia.edu/millionsong/challenge

widely adopted measures, MAP, AUC, and NDCG@10.

2 Related Work

Collaborative Filtering (CF) is arguably the most widely adopted technique in recommendation and has been applied to address both rating prediction and item recommendation tasks [Su and Khoshgoftaar, 2009]. There are two kinds of CF approaches, *i.e., memory-based* and *model-based* CF. Our proposed AdaBPR is based on model-based CF. Next, we first review model-based CF for item recommendation with implicit feedback and then review boosting techniques.

2.1 Item Recommendation with Implicit Feedback

Previous recommendation models are proposed mainly for rating prediction problems. The latent factor model implemented by matrix factorization is the most successful approach and has been applied to various rating prediction tasks [Bell and Koren, 2007; Wang *et al.*, 2013; Lu *et al.*, 2013; Hu *et al.*, 2014].

For item recommendation based on users' implicit feedback, model-based approaches aim to learn a personalized ranking function for each individual user. These approaches can be mainly categorized into three groups, *i.e.,* pointwise, pairwise, and listwise approaches, based on the loss functions used to learn the model parameters [Karatzoglou *et al.*, 2013]. The weighted regularized matrix factorization is a representative pointwise approach [Pan *et al.*, 2008; Hu *et al.*, 2008], which treats all observed entries in the user-item interaction matrix as positive examples and all missing entries as negative examples. Different confidence levels are assigned to the positive and negative examples. A least square loss function is then used to learn the latent features of users and items by approximating the pre-defined ratings. The PureSVD method [Cremonesi *et al.*, 2010] models each user as a combination of the item features. It does not need to learn user-specific parameters thus it can offer convenient optimization. The Sparse LInear Method (SLIM) [Ning and Karypis, 2011] employs a sparse linear model for item recommendation. In SLIM, the recommendation score for a new item is calculated as an aggregation of other items. For pairwise approaches, a common assumption is that a user prefers the items that she has interacted with than those items that she has not interacted with. The Bayesian Personalized Ranking (BPR) based methods [Rendle *et al.*, 2009; Pan and Chen, 2013] are representative pairwise approaches. Their objective is to rank the interacted items higher than the un-interacted items in the ranked list of items for each user. These approaches optimize the ranking metric AUC of the model on the tasks [Rendle *et al.*, 2009]. The listwise approaches aim to optimize the loss function defined on more complex ranking metrics. For example, the Tensor Factorization for MAP maximization (TFMAP) proposed in [Shi *et al.*, 2012a] optimizes the ranking metric MAP. It tries to create an optimally ranked item list for an individual user under a given context. Similarly, the Collaborative Less-is-More Filtering (CLiMF) model introduced in [Shi *et al.*, 2012b] explores the optimization of Mean Reciprocal Rank (MRR), which is a metric for measuring top-N item recommendation.

2.2 Boosting

Boosting is a general technique that can improve the accuracy of a given weak learning algorithm [Freund and Schapire, 1995]. The basic idea is to repeatedly construct a set of weak models using the weak learning algorithm on re-weighted training data. The weak models are then linearly combined to form a strong model, for better accuracy. The boosting technique was originally developed for the binary classification setting, where AdaBoost [Freund and Schapire, 1995] is the most well-known algorithm. It has also been extended to deal with other problems, including multi-class classification [Schapire and Singer, 1999], regression [Duffy and Helmbold, 2002], and ranking [Xu and Li, 2007].

Recently, the boosting technique has also been introduced to solve recommendation problems. Two boosting frameworks based on AdaBoost have been proposed to enhance the performances of CF approaches in rating prediction tasks [Jiang *et al.*, 2013]. Each framework consists of a set of homogeneous recommenders created by the same CF algorithm with different sample weights. The recent work [Cheng *et al.*, 2014] exploited the gradient boosting for context-aware recommendation. Specifically, a Gradient Boosting Factorization Machine (GBFM) model was proposed to select effective interaction features for rating prediction. The Adaptive Boosting Matrix Factorization (AdaMF) model in [Wang *et al.*, 2014] extended AdaRank for item recommendation, where the sample weights are assigned to individual users, and the component recommenders are constructed using the CF algorithm for rating prediction.

Among the exiting recommendation frameworks, the proposed AdaBPR is most related to AdaMF, but with two significant differences. First, in AdaBPR, a sample weight is assigned to each observed *user-item interaction*, but not an *individual user* as in AdaMF. Second, the component recommenders of AdaBPR are created by directly optimizing the weighted ranking metric (*i.e.,* AUC), not approximating users' ratings in AdaMF. Our empirical evaluations show that these two differences make AdaBPR much more effective for item recommendation tasks.

3 The Boosting Recommendation Framework

Given the historical interactions between m users and n items, the task of item recommendation is to recommend a target user u a list of items that she may prefer but has not interacted with before [Deshpande and Karypis, 2004]. In many real-world scenarios, the item recommendation tasks are based on users' implicit preference feedback, *e.g.,* the number of times that a user has interacted with an item. This kind of feedback is usually represented using a set of binary variables $y_{ui} \in \{0, 1\}$. If a user u has interacted with an item i, y_{ui} is set to 1, and 0 otherwise. Note that $y_{ui} = 0$ does not explicitly indicate that u is not interested in i. It may be the result that user u does not even know the existence of i. In this paper, we denote the set of users and items by U and V, respectively. For a user u, we denote her historical items by $V_u^+ = \{i|y_{ui} = 1, i \in V\}$ and define $V_u^- = V \setminus V_u^+$. Moreover, the set of all observed user-item interactions is denoted by $\mathcal{D} = \{(u, i)|u \in U, i \in V_u^+\}$.

For item recommendation tasks, the accuracy of a recommendation model is usually evaluated using the ranking metrics, *e.g.,* AUC, MAP, NDCG, Precision, and Recall. The definitions of AUC and MAP are given below.

$$\text{AUC} = \frac{1}{|U|}\sum_{u\in U}\frac{1}{|V_u^+|}\sum_{i\in V_u^+}\frac{1}{|V_u^-|}\sum_{j\in V_u^-}\mathbb{I}(\pi_{ui} < \pi_{uj}) \quad (1)$$

$$\text{MAP} = \frac{1}{|U|}\sum_{u\in U}\frac{\sum_{i\in V_u^+}\frac{\sum_{j\in V}\mathbb{I}(y_{uj}>0)\mathbb{I}(\pi_{uj}\leq\pi_{ui})}{\pi_{ui}}}{|V_u^+|} \quad (2)$$

where $|\mathcal{X}|$ denotes the cardinality of the set $\mathcal{X}$, π_{ui} is the rank position of i in the ranked item list that is created by sorting items according to u's preferences on them in descending order, and $\mathbb{I}(\cdot) = 1$ if the condition is true, and 0 otherwise.

3.1 AdaBPR

The proposed AdaBPR framework aims to optimize the loss function defined based on a personalized ranking metric.

Observe from Eq. 1 and 2 that the accuracy of a recommendation model is determined by the rank positions of the historical items V_u^+ of each user u. Thus, we use a general function $E[\pi(u, i, f)]$ to denote the ranking accuracy associated with each observed interaction pair (u, i) where $i \in V_u^+$. The argument of the general function $\pi(u, i, f)$ is the rank position of item i in the ranked item list of u, resulted by a learned ranking model f. Then, the ranking accuracy in terms of a ranking metric, *e.g.,* AUC or MAP, on the training data is rewritten as below, where $\beta_u = \frac{1}{|V_u^+|}$.

$$\frac{1}{|U|}\sum_{u\in U}\frac{1}{|V_u^+|}\sum_{i\in V_u^+}E[\pi(u,i,f)] \propto \sum_{(u,i)\in\mathcal{D}}\beta_u E[\pi(u,i,f)].$$

To maximize the ranking accuracy, we propose to minimize the following loss function:

$$\arg\min_{f\in\mathcal{F}}\sum_{(u,i)\in\mathcal{D}}\beta_u\{1 - E[\pi(u,i,f)]\}, \quad (3)$$

where $\mathcal{F}$ is the set of possible ranking functions. Observation that this minimization is equivalent to maximizing the performance measures (*e.g.,* AUC or MAP). Because E is a non-continuous function, it is difficult to optimize the loss function defined in Eq. 3. To solve this issue, we propose to minimize its upper bound as follows:

$$\arg\min_{f\in\mathcal{F}}\sum_{(u,i)\in\mathcal{D}}\beta_u\exp\{-E[\pi(u,i,f)]\}. \quad (4)$$

The primary idea of applying boosting for item recommendation is to learn a set of homogeneous component recommenders and then create an ensemble of the component recommenders to predict users' preferences. Here, we use a linear combination of component recommenders as the final recommendation model:

$$f = \sum_{t=1}^{T}\alpha_t h^{(t)}, \quad (5)$$

where $h^{(t)}$ is the t^{th} component recommender and α_t is a positive weight assigned to $h^{(t)}$ to determine its contribution in the final recommendation model. In the training process, AdaBPR runs for T rounds, and one component recommender is created at each round. At the t^{th} round, given the former $t-1$ component recommenders, the optimization problem in Eq. 4 is converted to

$$\arg\min_{\alpha_t, h^{(t)}\in\mathcal{H}}\sum_{(u,i)\in\mathcal{D}}\beta_u\exp\{-E[\pi(u,i,f^{(t-1)}+\alpha_t h^{(t)})]\}, \quad (6)$$

where $\mathcal{H}$ is the set of possible component recommenders and $f^{(t-1)} = \sum_{k=1}^{t-1}\alpha_k h^{(k)}$. To solve Eq. 6, we first create an optimal component recommender $h^{(t)}$ by using a re-weighting strategy, which assigns a dynamic weight $w_{ui}^{(t)}$ on each observed user-item interaction $(u, i) \in \mathcal{D}$. At each round, AdaBPR increases the weights of the observed user-item interactions that are not ranked well by the ensemble recommender created so far. The learning of the next component recommender will then pay more attention to those "hard" interactions. Suppose $h^{(t)}$ is given, its weight α_t can be solved. The details of the AdaBPR framework is presented in Alg. 1. A lower bound on the personalized ranking accuracy for AdaBPR is presented in Theorem 1. This bound is derived following similar techniques for AdaRank in [Xu and Li, 2007].

Algorithm 1: The AdaBPR Algorithm

Input : The observed user-item interactions $\mathcal{D}$, and parameters E and T

Output: The ensemble recommender $f^{(T)}$.

Initialize $w_{ui}^{(1)} = \beta_u / \sum_{(u,i)\in\mathcal{D}}\beta_u, \forall(u,i)\in\mathcal{D}$;

for $t = 1, \ldots, T$ **do**

  Create the component recommender $h^{(t)}$ using Alg. 2, based on $\mathcal{D}$ and the dynamic weights $w_{ui}^{(t)}, \forall(u,i)\in\mathcal{D}$;

  Compute the ranking accuracy measure $E[\pi(u,i,h^{(t)})], \forall(u,i)\in\mathcal{D}$;

  Compute the weight α_t assigned to $h^{(t)}$, $\alpha_t = \frac{1}{2}\ln\frac{\sum_{(u,i)\in\mathcal{D}}w_{ui}^{(t)}\{1+E[\pi(u,i,h^{(t)})]\}}{\sum_{(u,i)\in\mathcal{D}}w_{ui}^{(t)}\{1-E[\pi(u,i,h^{(t)})]\}}$;

  Create the ensemble recommender $f^{(t)}$, $f^{(t)} = \sum_{k=1}^{t}\alpha_k h^{(k)}$;

  foreach $(u,i)\in\mathcal{D}$ **do**

    $w_{ui}^{(t+1)} = \frac{\beta_u\exp\{-E[\pi(u,i,f^{(t)})]\}}{\sum_{(u,i)\in\mathcal{D}}\beta_u\exp\{-E[\pi(u,i,f^{(t)})]\}}$;

Theorem 1 *The following bound holds on the personalized ranking accuracy of the AdaBPR algorithm on training data:*

$$\frac{1}{|U|}\sum_{u\in U}\frac{1}{|V_u^+|}\sum_{i\in V_u^+}E[\pi(u,i,f)] \geq \frac{\sum_{(u,i)\in\mathcal{D}}\beta_u}{|U|}\left[1 - \prod_{t=1}^{T}e^{-\delta_{min}^t}\sqrt{1-\varphi(t)^2}\right],$$

where $\varphi(t) = \sum_{(u,i)\in\mathcal{D}}w_{ui}^{(t)}E[\pi(u,i,h^{(t)})]$, $\delta_{min}^t = \min_{(u,i)\in\mathcal{D}}\delta_{ui}^t$, *and* $\delta_{ui}^t = E[\pi(u,i,f^{(t-1)} + \alpha_t h^{(t)})] -$

Algorithm 2: Component Recommender Construction

Input : The observed interactions $\mathcal{D}$, the weight distribution $w_{ui}^{(t)}$, the learning rate η, and the regularization parameter λ.

Output: The learned user latent factors $\mathbf{P}^{(t)}$ and item latent factors $\mathbf{Q}^{(t)}$.

1 Initialize $\mathbf{P}^{(t)}$ and $\mathbf{Q}^{(t)}$ randomly;
2 **for** $e = 1, \ldots, max\_iter$ **do**
3 Randomly permutate the observed interactions $\mathcal{D}$;
4 **foreach** $(u, i) \in \mathcal{D}$ **do**
5 Randomly draw an item j from V_u^- ;
6 Update the model parameters θ as follows,
$$\theta \leftarrow \theta - \eta \left[\widehat{w}_{ui}^{(t)} \cdot \frac{\partial \ell(\Delta_{uij}^{(t)})}{\partial \Delta_{uij}^{(t)}} \cdot \frac{\partial \Delta_{uij}^{(t)}}{\partial \theta} + \lambda \theta \right],$$
where $\theta = \mathbf{p}_u^{(t)}$, $\mathbf{q}_i^{(t)}$, and $\mathbf{q}_j^{(t)}$;

$E[\pi(u, i, f^{(t-1)})] - \alpha_t E[\pi(u, i, h^{(t)})]$, *for all* $(u, i) \in \mathcal{D}$ *and* $t = 1, 2, \cdots, T$.

3.2 Constructing Component Recommender

To construct component recommenders, we adopt the latent factor based models, which are the most successful recommendation models in literature [Su and Khoshgoftaar, 2009]. The objective of latent factor model is to map both users and items into a shared latent space, with a low dimensionality $d \ll \min(|U|, |V|)$. Specifically, two latent vectors $\mathbf{p}_u^{(t)} \in \mathbb{R}^{1\times d}$ and $\mathbf{q}_i^{(t)} \in \mathbb{R}^{1\times d}$ are used to denote the properties of u and i respectively, in a component recommender $h^{(t)}$. The preference score of u to i is approximated by:

$$h_{ui}^{(t)} = \mathbf{p}_u^{(t)} \mathbf{q}_i^{(t)\top}. \tag{7}$$

Biases for users and items can also be incorporated in Eq. 7 to produce more accurate models. We further denote the latent factors of all users and all items by $\mathbf{P}^{(t)} \in \mathbb{R}^{|U|\times d}$ and $\mathbf{Q}^{(t)} \in \mathbb{R}^{|V|\times d}$ respectively.

At each round, the accuracy of the component recommender $h^{(t)}$ can be evaluated by the ranking performance measure E weighted by $w_{ui}^{(t)}$. The optimal $h^{(t)}$ is then obtained by consistently optimizing the weighted ranking measure. In our implementation, we choose AUC as the ranking metric to learn AdaBPR, considering both the effectiveness and efficiency factors. Note that AUC may be replaced by other ranking metrics whose range is within $[-1, +1]$, *e.g.,* MAP and NDCG [Xu and Li, 2007].

Given the weight distribution $w_{ui}^{(t)}$, the accuracy of the component recommender $h^{(t)}$, measured by weighted AUC, is defined as follows:

$$\begin{aligned} \text{wAUC} &= \sum_{(u,i)\in\mathcal{D}} w_{ui}^{(t)} \frac{1}{|V_u^-|} \sum_{j\in V_u^-} \mathbb{I}(\pi_{ui}^{(t)} < \pi_{uj}^{(t)}) \\ &= \sum_{(u,i)\in\mathcal{D}} w_{ui}^{(t)} \frac{1}{|V_u^-|} \sum_{j\in V_u^-} \mathbb{I}(h_{ui}^{(t)} > h_{uj}^{(t)}), \end{aligned} \tag{8}$$

Table 1: The statistics of the experimental datasets

Datasets	#users	#items	#user-item pairs
Movielens-100K	943	1,574	82,520
Movielens-1M	6,039	3,628	836,478
TPMSD-790K	27,216	9,994	789,915

where $\pi_{ui}^{(t)}$ denotes the rank position of the item i in the list ranked by $h^{(t)}$ for u. Maximizing of the weighted AUC is equivalent to minimizing the following loss function:

$$\min_{\mathbf{P}^{(t)},\mathbf{Q}^{(t)}} \sum_{(u,i)\in\mathcal{D}} w_{ui}^{(t)} \frac{1}{|V_u^-|} \sum_{j\in V_u^-} \mathbb{I}(h_{ui}^{(t)} \le h_{uj}^{(t)}). \tag{9}$$

To solve this problem, we replace the indicator function with a convex surrogate, *i.e.,* the hinge loss function, as follows:

$$\ell(\Delta_{uij}^{(t)}) = \max(0, 1 - \Delta_{uij}^{(t)}), \tag{10}$$

where $\Delta_{uij}^{(t)} = h_{ui}^{(t)} - h_{uj}^{(t)}$. The optimal component recommender $h^{(t)}$ is found by optimizing the following objective function:

$$\begin{aligned} \min_{\mathbf{P}^{(t)},\mathbf{Q}^{(t)}} & \sum_{(u,i)\in\mathcal{D}} \widehat{w}_{ui}^{(t)} \sum_{j\in V_u^-} \frac{1}{|V_u^-|} \ell(\Delta_{uij}^{(t)}) \\ & + \frac{\lambda}{2} \left(\left\| \mathbf{P}^{(t)} \right\|_F^2 + \left\| \mathbf{Q}^{(t)} \right\|_F^2 \right), \end{aligned} \tag{11}$$

where $\widehat{w}_{ui}^{(t)} = w_{ui}^{(t)} \cdot |\mathcal{D}|$, λ is a regularization parameter, and $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. The problem in Eq. 11 can be solved by stochastic gradient descent, detailed in Alg. 2. Note that in Line 6, the component $\frac{1}{|V_u^-|}$ does not appear in the updating of model parameter θ. The reason is that for a given observed interaction $(u, i) \in \mathcal{D}$, an item j is sampled from V_u^- with uniform probability $\frac{1}{|V_u^-|}$ to form a triplet (u, i, j) used to train the component recommender.

4 Experiments

In this section, we first empirically evaluate AdaBPR against state-of-the-art methods. We then study the impact of different parameter settings to the performances of AdaBPR.

4.1 Experimental Settings

Dataset Description

The experiments are conducted on three public datasets: the MovieLens-100K[2], MovieLens-1M, and the Taste Profile Subset of the Million Song Dataset (TPMSD) [Bertin-Mahieux *et al.*, 2011]. MovieLens-100K contains $100,000$ ratings given by 943 users to $1,682$ movies. MovieLens-1M contains $1,000,209$ ratings given by $6,040$ users to $3,952$ movies. For these two datasets, we keep the ratings larger than 3 as observed preference feedback, to simulate the implicit feedback scenario. Similar settings have been used in earlier studies [Pan and Chen, 2013]. The TPMSD contains $48,373,586$ listening records made by $1,019,318$ users over

[2]http://grouplens.org/datasets/movielens/

Table 2: The recommendation accuracies of PopRank, BPRMF, GBPR, AdaMF^{exp}, AdaMF^{imp}, and AdaBPR, measured by MAP, AUC, and NDCG@10. The best results are in **bold faces** and the second best results are underlined.

Datasets	Metrics	PopRank	BPRMF	GBPR	AdaMF^{exp}	AdaMF^{imp}	AdaBPR
Movielens-100k	MAP	0.1485	0.2780±0.0029	0.2907±0.0017	0.1296±0.0009	0.2893±0.0004	**0.3132**±0.0007
	AUC	0.8541	0.9320±0.0005	0.9338±0.0006	0.8288±0.0014	0.9322±0.0002	**0.9365**±0.0002
	NDCG	0.4962	0.6705±0.0005	0.6942±0.0031	0.4601±0.0076	0.6928±0.0037	**0.7310**±0.0040
Movielens-1M	MAP	0.1206	0.2172±0.0005	0.2295±0.0010	0.1097±0.0004	0.2286±0.0004	**0.2629**±0.0004
	AUC	0.8649	0.9277±0.0002	0.9315±0.0002	0.8458±0.0007	0.9318±0.0001	**0.9395**±0.0001
	NDCG	0.4875	0.6352±0.0005	0.6617±0.0022	0.4541±0.0009	0.6417±0.0015	**0.7124**±0.0015
TPMSD-790K	MAP	0.0466	0.0723±0.0012	0.0792±0.0017	N.A.	0.0735±0.0006	**0.1032**±0.0002
	AUC	0.8345	0.9048±0.0007	0.9065±0.0016	N.A.	0.9079±0.0004	**0.9190**±0.0001
	NDCG	0.1609	0.2624±0.0027	0.2795±0.0038	N.A.	0.2501±0.0021	**0.3474**±0.0007

$384,546$ songs. In our evaluation, a subset of TPMSD is used with the following sampling strategy. We first randomly sample $10,000$ songs that have been listened by at least 50 different users. We then extract the users who have listened 20 or more different sampled songs and their listening history over the sampled songs. As the result, we have about 790K interactions made by $27,216$ users to $9,994$ songs and we denote this dataset by TPMSD-790K. Table 1 summarizes the three datasets where the movies and songs are "items".

Setup and Metrics

Each dataset is randomly partitioned into two non-overlapping sets for training and testing. For each user u, 70% of her historical items V_u^+ are randomly chosen as training data, and the remaining 30% of V_u^+ are used for testing. The densities of the user-item interaction matrix generated from the training data of the three datasets MovieLens-100K, MovieLens-1M, and TPMSD-790K are 4.05×10^{-2}, 2.70×10^{-2}, and 2.08×10^{-3}, respectively.

The accuracy of a recommendation model is measured by using three metrics, namely MAP, AUC, and NDCG@10. The former two metrics are widely adopted for evaluation of ranking accuracy. Because users are usually interested in a few top-ranked items, NDCG@N is also used to compare the top-N recommendation performance. In our experiments, we set $N = 10$. For each metric, we first compute the accuracy for each user on the testing data, and then report the averaged accuracy over all users.

Evaluated Recommendation Algorithms

We evaluate AdaBPR and 4 baseline algorithms: PopRank, BPRMF, GBPR, and AdaMF.

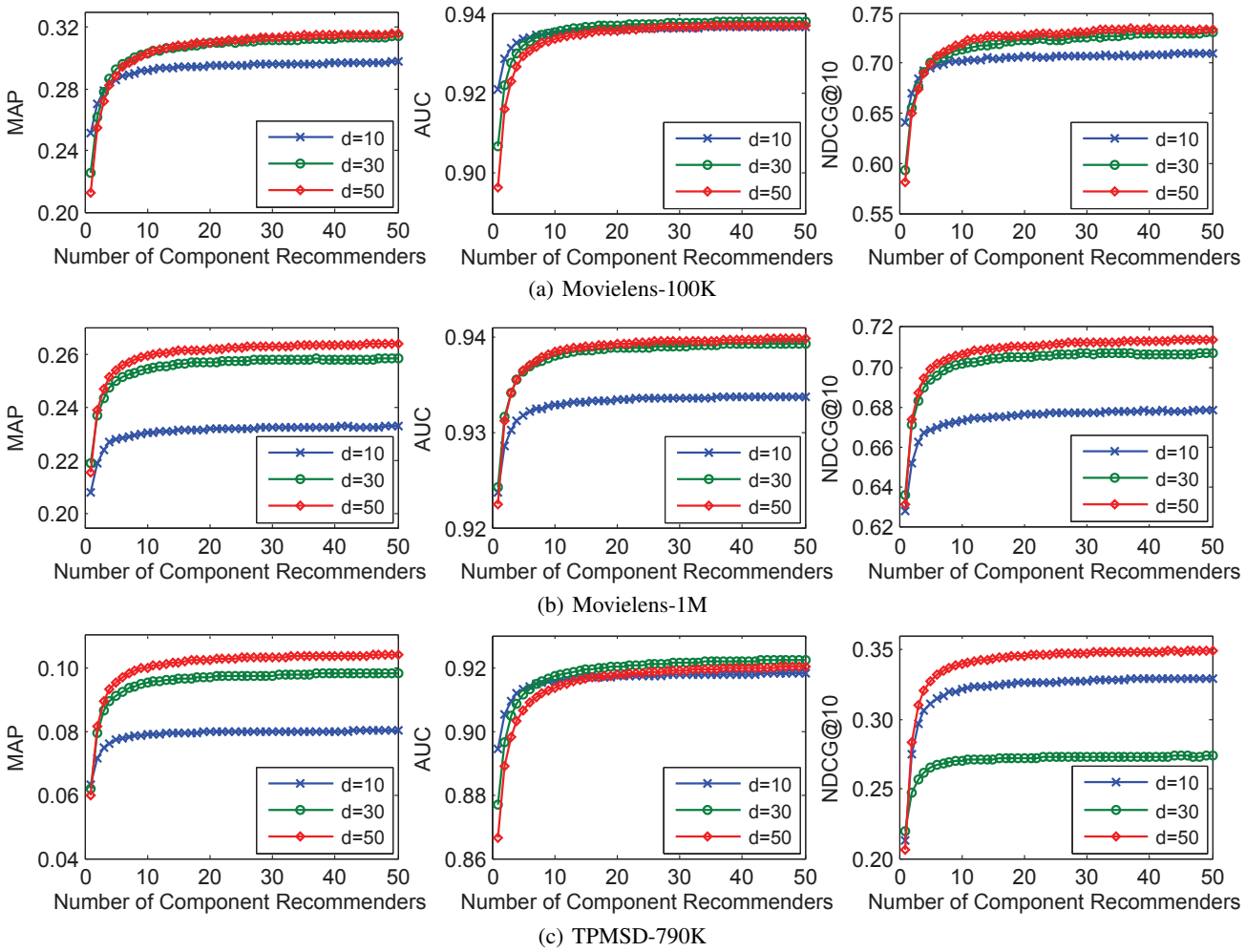
- PopRank is a naïve baseline that recommends items to users purely based on the popularity of items.
- BPRMF is a state-of-the-art approach designed for item recommendation with users' implicit feedback [Rendle *et al.*, 2009]. It is a pairwise approach and has outperformed pointwise methods [Pan *et al.*, 2008; Hu *et al.*, 2008], especially in terms of the ranking metric AUC.
- GBPR is an extension of BPRMF. It combines both the pairwise preference and group preference for item recommendation [Pan and Chen, 2013].
- AdaMF is a boosting approach that extends AdaRank for item recommendation [Wang *et al.*, 2014]. It generates item recommendations by approximating the explicit ratings (*e.g.,* 1-5) given by users. We denote this original AdaMF algorithm by AdaMF^{exp}. We also evaluated AdaMF^{imp} which is AdaMF with implicit feedback. The rating of an observed user-item interaction pair is set to 1 and the rating of an un-observed interaction pair is 0. For a user u with item history V_u^+, the same number of items are uniformed sampled from V_u^- to train the model.

We adopt cross-validation to choose the parameters for all the evaluated algorithms. Similar to that in [Pan and Chen, 2013], the validation data are constructed by randomly taking 1 historical item of each user from the training data. For AdaBPR, the dimensionality of the latent space d is selected from $\{10, 30, 50\}$ (see Section 3.2), and for the other models, d is chosen from $\{10, 30, 50, 70, 90\}$. The regularization parameters are chosen from $10^{[-6:1:0]}$ (see Eq. 11), and the optimal learning rates are selected from $2^{[-10:1:0]}$ (see Alg. 2). For GBPR, the group size is chosen from $\{2, 3, 4, 5\}$ and the parameter ρ is chosen from $\{0.2, 0.4, 0.6, 0.8, 1.0\}$. The number of component models in AdaMF^{exp}, AdaMF^{imp}, and AdaBPR are all set to 30. All experiments are repeated for 5 times, each with a different random seed. The results reported are average of the 5 runs.

4.2 Performance Evaluation

The recommendation accuracies of AdaBPR and other baseline methods are summarized in Table 2. We make the following observations:

- On all three datasets, the proposed AdaBPR algorithm significantly outperforms all the other baseline algorithms, measured by the three metrics. The improvements over the baselines are statistically significant with $p < 0.001$ using Wilcoxon signed rank significance test [Shani and Gunawardana, 2011].
- The proposed AdaBPR attains significant improvement over other pairwise approaches without boosting, *i.e.,* BPRMF and GBPR. For instance, in terms of NDCG@10, AdaBPR outperforms BPRMF and GBPR by 32.39% and 24.29%, respectively, on TPMSD-790K dataset. This result shows that the accuracy of item recommendation can be largely improved by using boosting technique.
- Compared with AdaMF^{imp}, AdaBPR achieves much better results on all datasets for all the metrics. For example,



(a) Movielens-100K

(b) Movielens-1M

(c) TPMSD-790K

Figure 1: Performance trend of AdaBPR measured by MAP, AUC, and NDCG@10. The number of the component recommenders ranges from 1 to 50, and the dimensionality of the latent space $d \in \{10, 30, 50\}$ in each component recommender.

in terms of NDCG@10, AdaBPR outperforms AdaMF^{imp} by 5.51%, 11.02%, and 38.90%, respectively, on the three datasets. This result clearly demonstrates the effectiveness of our proposed algorithm. The main reason is that the component recommenders in AdaBPR are learnt by using the algorithm designed for personalized ranking problems, while the component recommenders in AdaMF are built based on the algorithms for rating prediction tasks.

- AdaMF^{exp} performs much poorer than AdaMF^{imp} and other baseline methods. This suggests that AdaMF is more effective with implicit feedback than with explicit feedback, for item recommendation tasks.

To evaluate the sensitivity to parameters for AdaBPR, we study its performance with respect to different parameter settings. The number of component recommenders is adjusted from 1 to 50 and the dimensionality of the latent space d of each component recommender is varied in $\{10, 30, 50\}$. The results on all the three datasets are summarized in Figure 1, in terms of MAP, AUC, and NDCG@10. Observed that the ranking performance generally improves with the increase of number of component recommenders, particularly when the number of component recommenders are fewer than 10. When there are more than 10 component recommenders, introducing more component recommenders leads to marginal improvements. Regarding the dimensionality of latent space factor d, larger d generally achieves better results. The only exception is the NDCG@10 measure on TPSMD-790K dataset where d=10 leads to better result than $d = 30$. Nevertheless, $d = 50$ achieves the best results or the second best results in all the experiment settings.

4.3 Conclusion and Future Work

In this paper, we propose a novel boosting algorithm, named AdaBPR (Adaptive Boosting Personalized Ranking), for the item recommendation tasks with users' implicit feedback. AdaBPR creates an ensemble of a set of homogeneous component recommenders built by the same CF algorithm, to predict users' preferences on items. Empirical results on three datasets demonstrate the effectiveness of the proposed algorithm in comparison with a few strong baseline methods.

As a part of future work, we would like to construct component recommenders that directly optimize more complex ranking metrics, such as MAP [Shi *et al.*, 2012a] and MRR [Shi *et al.*, 2012b]. We are also interested in extending AdaBPR to build live recommender systems that can evolve on-the-fly over time. In this scenario, a component recommender may be built using the online AUC maximization (OAM) method proposed in [Zhao *et al.*, 2011], which can directly optimize the ranking metric AUC under the online learning settings.

Acknowledgements

This research is supported, in part, by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative and administered by the Interactive and Digital Media Programme Office (Grant No. MDA/IDM/2012/8/8-2 VOL 01). This research is also supported, in part, by the Competitive Research Grant from the National Research Foundation in Singapore (Grant No. NRF-CRP8-2011-05).

References

[Bell and Koren, 2007] Robert M Bell and Yehuda Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[Bertin-Mahieux *et al.*, 2011] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *ISMIR'11*, 2011.

[Chen *et al.*, 2009] Jilin Chen, Werner Geyer, Casey Dugan, Michael Muller, and Ido Guy. Make new friends, but keep the old: recommending people on social networking sites. In *CHI'09*, pages 201–210. ACM, 2009.

[Cheng *et al.*, 2014] Chen Cheng, Fen Xia, Tong Zhang, Irwin King, and Michael R. Lyu. Gradient boosting factorization machines. In *RecSys'14*, pages 265–272. ACM, 2014.

[Cremonesi *et al.*, 2010] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys'10*, pages 39–46. ACM, 2010.

[Deshpande and Karypis, 2004] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *TOIS*, 22(1):143–177, 2004.

[Duffy and Helmbold, 2002] Nigel Duffy and David Helmbold. Boosting methods for regression. *Machine Learning*, 47(2-3):153–200, 2002.

[Freund and Schapire, 1995] Yoav Freund and Robert E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[Hu *et al.*, 2008] Yifan Hu, Florham Park, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM'08*, 2008.

[Hu *et al.*, 2014] Longke Hu, Aixin Sun, and Yong Liu. Your neighbors affect your ratings: on geographical neighborhood influence to rating prediction. In *SIGIR'14*, pages 345–354. ACM, 2014.

[Jiang *et al.*, 2013] Xiaotian Jiang, Zhendong Niu, Jiamin Guo, Ghulam Mustafa, Zihan Lin, Baomi Chen, and Qian Zhou. Novel boosting frameworks to improve the performance of collaborative filtering. In *ACML'13*, pages 87–99, 2013.

[Karatzoglou *et al.*, 2013] Alexandros Karatzoglou, Linas Baltrunas, and Yue Shi. Learning to rank for recommender systems. In *RecSys'13*, pages 493–494. ACM, 2013.

[Linden *et al.*, 2003] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[Liu *et al.*, 2014] Yong Liu, Wei Wei, Aixin Sun, and Chunyan Miao. Exploiting geographical neighborhood characteristics for location recommendation. In *CIKM'14*, pages 739–748. ACM, 2014.

[Lu *et al.*, 2013] Jing Lu, Steven Hoi, Jialei Wang, and Peilin Zhao. Second order online collaborative filtering. In *ACML'13*, pages 325–340, 2013.

[Ning and Karypis, 2011] Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. In *ICDM'11*, pages 497–506. IEEE, 2011.

[Pan and Chen, 2013] Weike Pan and Li Chen. Gbpr: Group preference based bayesian personalized ranking for one-class collaborative filtering. In *IJCAI'13*, pages 2691–2697. AAAI Press, 2013.

[Pan *et al.*, 2008] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *ICDM'08*, pages 502–511. IEEE, 2008.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI'09*, pages 452–461, 2009.

[Schapire and Singer, 1999] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999.

[Shani and Gunawardana, 2011] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[Shi *et al.*, 2012a] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Alan Hanjalic, and Nuria Oliver. Tfmap: optimizing map for top-n context-aware recommendation. In *SIGIR'12*, pages 155–164. ACM, 2012.

[Shi *et al.*, 2012b] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *RecSys'12*, pages 139–146. ACM, 2012.

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

[Wang *et al.*, 2013] Jialei Wang, Steven C.H. Hoi, Peilin Zhao, and Zhi-Yong Liu. Online multi-task collaborative filtering for on-the-fly recommender systems. In *RecSys'13*, pages 237–244. ACM, 2013.

[Wang *et al.*, 2014] Yanghao Wang, Hailong Sun, and Richong Zhang. Adamf: Adaptive boosting matrix factorization for recommender system. In *WAIM'14*, pages 43–54. Springer, 2014.

[Xu and Li, 2007] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *SIGIR'07*, pages 391–398. ACM, 2007.

[Zhao *et al.*, 2011] Peilin Zhao, Rong Jin, Tianbao Yang, and Steven C.H. Hoi. Online auc maximization. In *ICML'11*, pages 233–240, 2011.