MRLR: Multi-level Representation Learning for Personalized Ranking in Recommendation

Zhu Sun[1*], **Jie Yang**[2*], **Jie Zhang**[1], **Alessandro Bozzon**[2], **Yu Chen**[1], **Chi Xu**[3]
[1]Nanyang Technological University, Singapore
[2]Delft University of Technology, The Netherlands
[3] Singapore Institute of Manufacturing Technology, Singapore
[1]{sunzhu,zhangj,chenyu}@ntu.edu.sg, [2]{j.yang-3,a.bozzon}@tudelft.nl, [3]cxu@simtech.a-star.edu.sg

Abstract

Representation learning (RL) has recently proven to be effective in capturing local item relationships by modeling item co-occurrence in individual user's interaction record. However, the value of RL for recommendation has not reached the full potential due to two major drawbacks: 1) recommendation is modeled as a rating prediction problem but should essentially be a personalized ranking one; 2) multi-level organizations of items are neglected for fine-grained item relationships. We design a unified Bayesian framework MRLR to learn user and item embeddings from a multi-level item organization, thus benefiting from RL as well as achieving the goal of personalized ranking. Extensive validation on real-world datasets shows that MRLR consistently outperforms state-of-the-art algorithms.

1 Introduction

Recommendation is a fundamental task on the Web to mitigate the information overload problem [Su and Khoshgoftaar, 2009]. Recently, representation learning (RL) has attracted a considerable amount of interest from various domains, with recommender systems being no exception [Grbovic *et al.*, 2015; Liang *et al.*, 2016; Vasile *et al.*, 2016; Covington *et al.*, 2016; Barkan and Koenigstein, 2016]. The popularization of RL in recommendation can be mainly attributed to the word embedding techniques (e.g. CBOW and Skip-gram [Mikolov *et al.*, 2013a; 2013b]) originated from the natural language processing (NLP) domain. Word embedding generally refers to the low-dimensional distributed representation of words [Bengio *et al.*, 2003], capturing syntactical and semantic relationships among words. The fast development of RL has enabled a series of methods for NLP tasks, among which the most significant are the extensions of word embedding to learn textual representations in different levels of granularity (e.g. document or paragraph RL [Le and Mikolov, 2014]), so as to help capture richer relationships between words and paragraphs or documents.

In recommendation, RL is used to capture local relationships between items, thus being called item embedding. Item embedding learns low-dimensional item representation by modeling item co-occurrence in individual user's interaction record, thus boosting recommendation accuracy. While it helps learn better item representation, item embedding alone (e.g. Item2Vec [Barkan and Koenigstein, 2016], CoFactor [Liang *et al.*, 2016], Meta-Prod2Vec [Vasile *et al.*, 2016]) does not allow for personalized recommendation. Inspired by document RL (e.g. PV-DM [Le and Mikolov, 2014]), an important branch of work explores the potential of item embedding in personalized recommendation by learning representations for both users and items – as documents and words respectively in NLP (e.g. User2Vec [Grbovic *et al.*, 2015]).

We argue that the potential of RL for recommendation has not been fully exploited. Two major aspects have been largely neglected: 1) recommendation is essentially a personalized ranking problem, while existing RL methods only model it as a rating prediction problem; 2) existing methods all ignore the possible multi-level organizations of items for uncovering fine-grained item relationships in recommendation (similar as word-paragraph-document in NLP), which could in turn help achieve better personalized ranking performance.

Personalized Ranking. It has proven that recommendation is better modeled as a personalized ranking problem [Weimer *et al.*, 2007; Rendle *et al.*, 2009]. Existing RL methods, however, optimize towards predicting user preferences over individual items (i.e. rating prediction), instead of predicting user preferences over a list of items (i.e. personalized ranking).

We therefore advocate for a RL method specifically designed for personalized ranking. It is however non-trivial to adapt item embedding to personalized ranking. The original item embedding method only learns from item co-occurrence relationships, whereas for personalized ranking the method has to learn from user-specific lists of items ranked w.r.t. user preferences. We hence first extend the original embedding method to a more generic Bayesian framework, under which we then fuse the likelihood function of user-specific pairwise item ranking. This unified framework can then learn user and item embedding from both item co-occurrence relationships and user-specific ranked lists of items, benefiting from user and item RL while reaching the goal of personalized ranking.

Multi-level RL. To fully exploit RL for better recommendation, we further extend the personalized ranking framework to multi-level RL, so as to capture fine-grained item relationships. Our method is inspired by paragraphs in NLP as the

*The first two authors contribute equally.

intermediate level of word organization between individual words and documents. Intuitively, each paragraph conveys a key message, and all the words in the paragraph helps support such message. Analogously , we introduce item categories as the intermediate level of item organization between individual items and items rated by the same user, since items with the same category often share similar characteristics. For example, online products are often described by categories as metadata such as clothing, books, electronics, and so on.

Our unified Bayesian framework therefore facilitates multi-level RL by combining RL in all the three levels (i.e. individual item, item category, and user). Although item category has recently been intensively studied [He *et al.*, 2016; Yang *et al.*, 2016], we are the first to investigate it from the perspective of multi-level RL, which enables our framework to capture the relationships of items in local context (i.e. item co-occurrence relationships), in the same category, and in user-specific ranked item list.

Original Contributions. Overall, we contribute a multi-level RL method for personalized ranking based recommendation (MRLR). To the best of our knowledge, we are the first to adopt RL for personalized ranking; meanwhile, we design multi-level RL to capture fine-grained item relationships by leveraging category RL as the intermediate level RL between item RL and user RL, thus to further enhance recommendation performance. Extensive validation on multiple real-world datasets shows that MRLR can consistently outperform state-of-the-art methods, resulting in a 5.18% lift in AUC.

2 Related Work

Rating prediction vs. personalized ranking

Recommendation is typically formulated as either a rating prediction problem or a personalized ranking one [Weimer *et al.*, 2007; Steck, 2013]. Personalized ranking has proven to be more direct and efficient than rating prediction, as most recommendations in real-world scenarios are presented in a ranked item list. In general, the rating prediction based algorithms estimate user preferences towards individual items as absolute scores, based on which items are ordered and recommended to users. Typical methods include probabilistic matrix factorization (PMF) [Mnih and Salakhutdinov, 2007], tensor factorization (TF) [Karatzoglou *et al.*, 2010] and factorization machine (FM) [Rendle, 2010]. In contrast, ranking based algorithms directly optimize towards learning users' preferences as personalized ranking on a set of items. Typical methods include CofiRank [Weimer *et al.*, 2007], BPR [Rendle *et al.*, 2009], CLiMF [Shi *et al.*, 2012].

Latent factor model vs. representation learning

State-of-the-art methods for recommendation are dominated by the latent factor model (LFM) [Shi *et al.*, 2014], which maps the high-dimensional user-item interaction matrix to low-dimensional latent user and item matrices. LFM based methods include all the representative rating prediction and ranking based methods mentioned above; in addition, many other effective methods fall into this category, such as NMF [Lee and Seung, 2001], CMF [Singh and Gordon, 2008], SVDFeature [Chen *et al.*, 2012] and SVD++ [Koren, 2008]. While these methods leverage global statistical information of user-item interaction data, they cannot capture fine-grained regularities in the latent factors [Pennington *et al.*, 2014].

Recently, representation learning (RL) based methods have drawn much attention. In contrast to LFM based methods, RL based approaches have shown to be highly effective in capturing local item relationships by modeling item co-occurrence in individual user's interaction record [Grbovic *et al.*, 2015; Barkan and Koenigstein, 2016; Liang *et al.*, 2016]. These methods are mostly inspired by the word embedding techniques, which can be traced back to the classical neural network language model [Bengio *et al.*, 2003], and the recent breakthrough of Word2Vec techniques, including CBOW and Skip-gram [Mikolov *et al.*, 2013a; 2013b]).

Representation learning in recommendation

Several RL based methods have been proposed to date. For example, Barkan and Koenigstein [2016] propose a neural item embedding model (Item2Vec) for collaborative filtering, which is capable of inferring item-to-item relationships. Vasile *et al.* [2016] extend Item2Vec to a more generic model by utilizing side-information to help compute the low-dimensional embeddings of items. However, they all fail to provide personalized recommendation, as embedding techniques are only used to learn better item representation. Several studies extend RL for personalization. Grbovic *et al.* [2015] first introduce the User2Vec model, which simultaneously learns representations of items and users by considering the user as a "global context". Liang *et al.* [2016] propose the CoFactor model, which jointly decomposes the user-item interaction matrix and the item-item co-occurrence matrix – equivalent to item embedding [Levy and Goldberg, 2014] – with shared item latent factors. However, all these methods model recommendation as a rating prediction problem.

In contrast, we propose a RL based method by formulating recommendation as personalized ranking. Furthermore, we consider multi-level RL, which can capture fine-grained item relationships in multi-level item organizations, to fully exploit RL for better personalized ranking performance.

3 The Proposed MRLR Framework

This section first formalizes recommendation as a personalized ranking problem, and then presents the multi-level RL framework (MRLR) to achieve the goal of personalized ranking, followed by the model learning methods.

3.1 Problem Formulation and Objective Function

Suppose we have m users $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$, and n items $\mathcal{I} = \{v_1, v_2, \ldots, v_n\}$. We use the binary user feedback matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$. If the interaction (rating) from u_p to v_i is observed, indicating u_p prefers v_i, then $\mathbf{R}_{pi} = 1$; otherwise 0. $\mathcal{I}^+_{u_p}$ is the set of items that user u_p prefers. $\mathcal{D}_r = \{(u_p, v_i, v_j) | u_p \in \mathcal{U}, v_i \in \mathcal{I}^+_{u_p}, v_j \in \mathcal{I} \backslash \mathcal{I}^+_{u_p}\}$ is the set of user-specific ranking triples indicating u_p prefers v_i to v_j, where $\mathcal{I} \backslash \mathcal{I}^+_{u_p}$ denotes the set of items that u_p has no interaction with. $\mathcal{D}_c = \{(u_p, v_i, v_k) | u_p \in \mathcal{U}, v_i, v_k \in \mathcal{I}^+_{u_p}\}$ is the set of item co-rated triples indicating u_p prefers both v_i and v_k. For each user, we aim to provide a personalized ranking list of items that she has not interacted with.

More specifically, our goal is to design a unified multi-level RL framework (MRLR) to learn user and item embeddings from both item co-rated relationships and user-specific ranked lists of items, thus to benefit from user and item RL, as well as to reach the goal of personalized ranking. We define the objective function of MRLR using a Bayesian framework, maximizing the following posterior probability,

$$P(\Theta|\mathcal{D}) \propto P(\mathcal{D}|\Theta)P(\Theta) \propto P(\mathcal{D}_c, \mathcal{D}_r|\Theta)P(\Theta) \quad (1)$$

where Θ is the set of parameters in MRLR, $\mathcal{D}$ is the observed data. It is proportional to maximizing the likelihood of the observed triples given the embeddings, i.e., $P(\mathcal{D}|\Theta)$. We define the likelihood function as the joint probability of item co-rated triples and user-specific ranking triples, i.e., $P(\mathcal{D}_c, \mathcal{D}_r|\Theta)$. Assuming the item co-rated triples and user-specific ranking triples are conditionally independent, the joint probability is then reformulated as follows:

$$\begin{aligned} &P(\mathcal{D}_c, \mathcal{D}_r|\Theta) = P(\mathcal{D}_c|\Theta)P(\mathcal{D}_r|\Theta) \\ &= \prod_{(u_p,v_i,v_k)\in\mathcal{D}_c} P((u_p, v_i, v_k)|\Theta) \prod_{(u_p,v_i,v_j)\in\mathcal{D}_r} P((u_p, v_i, v_j)|\Theta) \end{aligned} \quad (2)$$

where $P((u_p, v_i, v_k)|\Theta), P((u_p, v_i, v_j)|\Theta)$ denote the conditional probability of item co-rated triples and user-specific ranking triples, respectively. Hence, the MRLR framework seamlessly fuses the two components: (1) item co-rated triples for better user and item embedding; (2) user-specific ranking triples for personalized ranking. Besides, through multi-level RL, MRLR can fully exploit RL from a multi-level item organization, i.e., items in user-specific ranked list, items in a same category, and individual items, to capture fine-grained item relationships for better recommendation.

3.2 Modeling User and Item Embedding

For each user u_p and item $v_i \in \mathcal{I}^+_{u_p}$, the Skip-gram method [Mikolov *et al.*, 2013a; 2013b] aims at predicting the probability of item $v_k \in \mathcal{I}, (i \neq k)$ also preferred by u_p, i.e. $P(v_k|v_i)$, which is calculated by the softmax function:

$$P(v_k|v_i, \Theta) = \frac{exp(\mathbf{v}_i^T \mathbf{v}'_k)}{\sum_{v_g\in\mathcal{I}} exp(\mathbf{v}_i^T \mathbf{v}'_g)} \quad (3)$$

where $\mathbf{v}_i, \mathbf{v}_k, \mathbf{v}_g$ are embeddings of items v_i, v_k, v_g.

To allow for personalization, we model user u_p's preference towards item v_k by a similar softmax function:

$$P(v_k|u_p, \Theta) = \frac{exp(\mathbf{u}_p^T \mathbf{v}'_k)}{\sum_{v_g\in\mathcal{I}} exp(\mathbf{u}_p^T \mathbf{v}'_g)} \quad (4)$$

where $\mathbf{u}_p$ denotes the user embedding of u_p.

We now model the item co-rated triples $P((u_p, v_i, v_k)|\Theta)$. It should properly accommodate both the item co-rated relationships (Eq.3), and personalization (Eq.4). Instead of directly optimizing $P((u_p, v_i, v_k)|\Theta)$, we optimize the conditional probability $P(v_k|(u_p, v_i), \Theta)$, $P(v_i|(u_p, v_k), \Theta)$ and $P(u_p|(v_i, v_k), \Theta)$. Since we aim to recommend items to given users, we do not need to model $P(u_p|(v_i, v_k), \Theta)$. We take $P(v_k|(u_p, v_i)\Theta)$ for example. Inspired by document RL in NLP [Le and Mikolov, 2014], the user and item embeddings $\mathbf{u}_p, \mathbf{v}_i$ are summed as the new condition to predict the probability of v_k rated by u_p, given by,

$$P(v_k|(u_p, v_i), \Theta) = \frac{exp(\alpha_1 \mathbf{u}_p^T \mathbf{v}'_k + \alpha_2 \mathbf{v}_i^T \mathbf{v}'_k)}{\sum_{v_g\in\mathcal{I}} exp(\alpha_1 \mathbf{u}_p^T \mathbf{v}'_g + \alpha_2 \mathbf{v}_i^T \mathbf{v}'_g)} \quad (5)$$

where $\alpha_1 + \alpha_2 = 1.0$; $exp(\alpha_1 \mathbf{u}_p^T \mathbf{v}'_k + \alpha_2 \mathbf{v}_i^T \mathbf{v}'_k)$ aims to take into account both the personalized aspect by the term $\mathbf{u}_p^T \mathbf{v}'_k$, and item co-rated relationships by the term $\mathbf{v}_i^T \mathbf{v}'_k$. We model $P(v_i|(u_p, v_k), \Theta)$ in a similar way.

3.3 Modeling Personalized Ranking

We now proceed to model the user-specific ranking triples $P((u_p, v_i, v_j)|\Theta)$, to achieve the goal of personalized ranking. Similarly, we optimize the conditional probability of $P((v_j, v_i)|u_p, \Theta)$ and $P(u_p|(v_j, v_i), \Theta)$ instead of $P((u_p, v_i, v_j)|\Theta)$. As our goal is to recommend items, we only consider $P((v_j, v_i)|u_p, \Theta)$, which involves a user's preference over a pair of items. Based on Eq.4, we further deduce a user's preference on a pair of items. As the triple (u_p, v_i, v_j) indicates that u_p prefers v_i to v_j, it means that for u_p, we should maximize the probability that v_i is preferred by u_p but v_j is not favored by u_p. We denote such probability by $P((\neg v_j, v_i)|u_p, \Theta)$, which is defined as below:

$$P((\neg v_j, v_i)|u_p, \Theta) = \frac{exp(\mathbf{u}_p^T \mathbf{v}'_i - \mathbf{u}_p^T \mathbf{v}'_j)}{\sum_{v_h,v_g\in\mathcal{I}} exp(\mathbf{u}_p^T \mathbf{v}'_h - \mathbf{u}_p^T \mathbf{v}'_g)} \quad (6)$$

where the term $exp(\mathbf{u}_p^T \mathbf{v}'_i - \mathbf{u}_p^T \mathbf{v}'_j)$ denotes the preference difference of user u_p towards items v_i and v_j.

3.4 Modeling Multi-level Item Organization

We further consider multi-level granularity of item organizations to capture fine-grained item relationships. Specifically, we introduce item category as the intermediate level between items in the same user-specific ranked list and individual items. The rationale behind is that items in a same category generally share similar characteristics.

To integrate the influence of item category for better recommendation, we extend our framework to multi-level RL. The item embedding is thus reformulated as,

$$\overline{\mathbf{v}}_i = \mathbf{v}_i + \frac{\alpha_3}{|\mathcal{C}_{v_i}|} \sum_{c_l\in\mathcal{C}_{v_i}} \mathbf{c}_l \quad (7)$$

where $\mathcal{C}_{v_i}$ is the set of categories that v_i belongs to; $|\mathcal{C}_{v_i}|$ is the size of $\mathcal{C}_{v_i}$; $\mathbf{c}_l$ is the embedding for category c_l. By replacing the item embedding in Eq.5 and 6, the category RL can adapt item embedding, serving as the intermediate level RL. MRLR can now capture fine-grained relationships of items in local context (i.e., item co-rated relationships), in the same category, and in user-specific ranked item list.

3.5 Model Learning

Optimizing our MRLR framework is proportional to minimizing the negative log-likelihood function, given by,

$$\begin{aligned} \min_{\Theta} \mathcal{J} = &- \sum_{(u_p,v_i,v_k)\in\mathcal{D}_c} logP((u_p, v_i, v_k)|\Theta) - \\ &\sum_{(u_p,v_i,v_j)\in\mathcal{D}_r} logP((u_p, v_i, v_j)|\Theta) + \lambda_\Theta \Omega(\Theta) \end{aligned} \quad (8)$$

Algorithm 1: The optimization of MRLR

Input: $\mathbf{R}, \mathcal{C}, \lambda_\Theta, \alpha, \gamma, d, iter$
1 Initialize $\Theta = \{\mathbf{u}, \mathbf{v}, \mathbf{c}\}$ with small values;
2 Randomly sample (u_p, v_i, v_j) for $\mathcal{D}_r$;
// Negative sampling procedure
3 **foreach** $(u_p, v_i, v_k) \in \mathcal{D}_c$, *and* $(u_p, v_i, v_j) \in \mathcal{D}_r$ **do**
4 Draw N negative instances from the distribution $P(\mathcal{D}_c^-)$;
5 Draw N negative instances from the distribution $P(\mathcal{D}_r^-)$;
// Parameter update
6 **for** $t = 1; t \leq iter; t++$ **do**
7 **foreach** $(u_p, v_i, v_k) \in \mathcal{D}_c$, *and* $(u_p, v_i, v_j) \in \mathcal{D}_r$ **do**
8 $\mathbf{u}_p^{(t)} \leftarrow \mathbf{u}_p^{(t-1)} - \gamma \nabla \mathcal{J}(\mathbf{u}_p)$;
9 $\mathbf{v}^{(t)} \leftarrow \mathbf{v}^{(t-1)} - \gamma \nabla \mathcal{J}(\mathbf{v}), \mathbf{v}=\{\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k, \mathbf{v}_g, \mathbf{v}_h\}$;
10 **for** $l = 1; l \leq |\mathcal{C}_v|; l++$ **do**
11 $\mathbf{c}_l^{(t)} \leftarrow \mathbf{c}_l^{(t-1)} - \gamma \nabla \mathcal{J}(\mathbf{c}_l)$;
12 **if** $\mathcal{J}$ *has converged* **then**
13 break;

where $\Omega(\Theta)$ is the regularizer to prevent over-fitting, and λ_Θ is the regularization coefficient. To solve the optimization problem, we apply the stochastic gradient descent (SGD) method to the objective function $\mathcal{J}$.

Approximation of softmax function

It is impractical to directly adopt the softmax functions $P(v_k|(u_p, v_i), \Theta)$, $P(v_i|(u_p, v_k), \Theta)$ and $P((\neg v_j, v_i)|u_p, \Theta)$ to optimize our framework, since the cost of computing the denominators of these functions is proportional to the total number of items (n), which is considerably huge in real-world applications. To accelerate the speed, we adopt negative sampling proposed in [Mikolov *et al.*, 2013b]. Take $P(v_k|(u_p, v_i), \Theta)$ as an example, which can be approximated via negative sampling as follows:

$$P(v_k|(u_p, v_i), \Theta) = \sigma(\mathbf{u}_p^T \mathbf{v}'_k + \mathbf{v}_i^T \mathbf{v}'_k) \prod_{g=1}^{N} \mathbb{E}_{(u_p, v_i, v_g) \sim P(\mathcal{D}_c^-)} \sigma(-(\mathbf{u}_p^T \mathbf{v}'_g + \mathbf{v}_i^T \mathbf{v}'_g)) \quad (9)$$

where $\sigma(x) = 1/(1 + exp(-x))$ is the sigmoid function; $\mathcal{D}_c^- = \mathcal{D}_r$ is the opposite triple set of $\mathcal{D}_c$; $P(\mathcal{D}_c^-)$ is a function randomly sampling instances from $\mathcal{D}_c^-$. N is the number of negative instances to be drawn per positive instance. The idea behind negative sampling is that we want to maximize the similarity between v_k and (u_p, v_i) and minimize the similarity between a randomly sampled item v_g and (u_p, v_i). In this way, we can approximately maximize $P(v_k|(u_p, v_i), \Theta)$.

Similarly, $P(v_i|(u_p, v_k), \Theta), P((\neg v_j, v_i)|u_p, \Theta)$ are also approximated via negative sampling. One issue we should deal with is that computing the numerators of the softmax function $P((\neg v_j, v_i)|u_p, \Theta)$ is also very expensive, as we have at least $\mathcal{O}(mn * \min(|\mathcal{I}_{u_1}^+|, \cdots, |\mathcal{I}_{u_m}^+|))$ training triples in $\mathcal{D}_r$, where $|\mathcal{I}_{u_m}^+|$ is the size of $\mathcal{I}_{u_m}^+$. We thus randomly sample user-specific ranking triples instead of using all the triples. The optimization process is shown in Algorithm 1, which is mainly composed of two steps, i.e., negative sampling (line 3-5), and parameter update (line 6-13).

Complexity analysis

The computational time is mainly taken by evaluating the objective function $\mathcal{J}$ and updating the related variables. The time to compute $\mathcal{J}$ is $\mathcal{O}\big(d|\mathcal{D}_c| + d|\mathcal{D}_r|\big)$, where d is the dimension of embeddings, and $|\mathcal{D}_c|, |\mathcal{D}_r|$ are the sizes of item co-rated triples and user-specific ranking triples, respectively. For all gradients $\nabla\mathcal{J}(\mathbf{u}_p), \nabla\mathcal{J}(\mathbf{v}_i), \nabla\mathcal{J}(\mathbf{c}_l)$, the computational time are $\mathcal{O}\big(d|\mathcal{D}_c| + d|\mathcal{D}_r|\big)$, $\mathcal{O}\big(d|\mathcal{D}_c| + d|\mathcal{D}_r|\big)$ and $\mathcal{O}\big(d(|\mathcal{D}_c| + |\mathcal{D}_r|)|\mathcal{C}_{v_i}|\big)$, respectively. $|\mathcal{C}_{v_i}|$ is generally no larger than 10 in real-world applications [Yang *et al.*, 2016]. Hence, the overall computational complexity is $(\#iteration * \mathcal{O}(d|\mathcal{D}_c| + d|\mathcal{D}_r|))$. Specifically, $|\mathcal{D}_c| \leq mq(q-1)/2$, where $q = \max(|\mathcal{I}_{u_1}^+|, \cdots, |\mathcal{I}_{u_m}^+|)$. In real-world, q is typically small (e.g., power-law distribution). For $\mathcal{D}_r$, as illustrated before, we adopt the random sampling method to reduce its number. To sum up, MRLR is scalable to large datasets.

4 Experiments and Results

4.1 Experimental Setup

Datasets. We adopt the Amazon Web store data [McAuley *et al.*, 2015], which contains a series of datasets from various domains (e.g., clothing, electronics). To evaluate the effectiveness of MRLR, we choose four datasets, including Clothing, Electronics, Sports, Home. Besides user-item interactions, the datasets also include the categories that each item belongs to. We uniformly sample the datasets, to balance their sizes in the same order of magnitude for cross-dataset comparison. Table 1 reports the statistics of the datasets.

Table 1: Statistics of the datasets.

Datasets	#Users	#Items	#Ratings	#Categories
Clothing	29,550	50,677	181,993	1,764
Electronics	59,457	64,348	518,291	1,292
Sports	28,708	46,315	237,578	1,293
Home	37,884	50,948	313,871	2,002

Comparison Methods. We compare with seven state-of-the-art algorithms, 1) PMF [Mnih and Salakhutdinov, 2007]: probabilistic matrix factorization; 2) BPR [Rendle *et al.*, 2009]: Bayesian personalized ranking; 3) FM [Rendle, 2010]: factorization machine fusing item category. We only compare with FM, as it generally outperforms other LFM based methods; 4) Item2Vec [Barkan and Koenigstein, 2016]: item embedding based method; 5) Meta-Prod2Vec [Vasile *et al.*, 2016]: fuses item category based on Item2Vec; 6) CoFactor [Liang *et al.*, 2016]: jointly factorizes rating and item co-rated matrices; 7) User2Vec [Grbovic *et al.*, 2015]: considers the user as a global context while learning item embedding; Besides, four variants of our framework are compared, a) RL: RL model only considering user and item embedding ; b) PR: personalized ranking model; c) RLR: the RL model combining a) and b); d) MRLR: multi-level RL model with multi-level item organizations based on c).

Evaluation. Standard 5-fold cross validation is adopted to evaluate all the methods. The Area Under the ROC Curve (AUC) is used as the evaluation metric. Larger AUC indicates better recommendation performance.

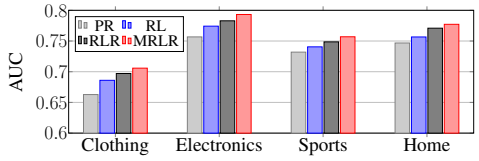

Figure 1: The results of our four variants.

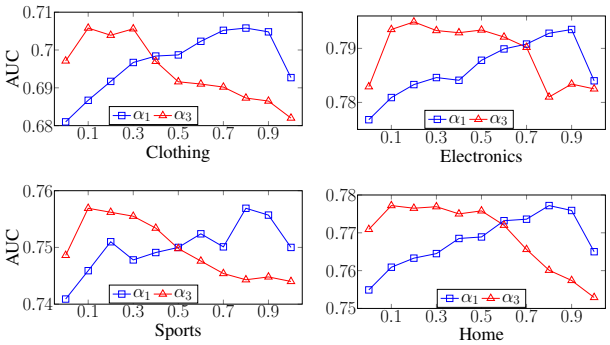

Figure 2: The effects of parameters α_1, α_3.

Parameter Settings. We empirically find out the optimal parameter settings for all method. We set $d = 10$. We apply a grid search in $\{0.001, 0.01, 0.1, 1.0\}$ for the learning rate γ, λ_Θ and 1/2-way regularization of FM, and a grid search in $\{1, 5, 10, 20, 50\}$ for the number of negative instances N.

4.2 Results of MRLR

Results of Variants. The performance of our four variants is depicted by Figure 1. RLR outperforms both PR and RL by 3.54% and 1.42% in AUC respectively (both significant, Paired t-test with p-value $< .01$), showing the effectiveness of both representation learning and personalized ranking. MRLR combining RLR with multi-level item organizations, performs the best among the four variants – with 1.12% lift in AUC compared to RLR (p-value $< .01$), indicating the benefit of considering fine-grained item relationships.

Impacts of Parameter α. Parameters α_1, α_2 control the importance of personalization and item co-occurrence relationships as shown in Eq.5. α_3 controls the effect of item category for adapting item embedding as shown in Eq.7. We apply a grid search ranging from 0 to 1 with step 0.1 to investigate their impacts. As $\alpha_1 + \alpha_2 = 1$, we only study the impacts of α_1, α_3, and we fix one and vary the other each time. The results are described by Figure 2. For the four datasets, as α_1 varies from small to large, the performance first increases then decreases, with the maximum reached at around 0.8. This indicates that user preferences play an important role in item recommendation. In terms of α_3, we observe that the optimal settings range from 0.1 to 0.2, denoting a substantial contribution of item category in recommendation. The similarity in performance variation across α_1, α_3 values on the four datasets demonstrates the robustness of MRLR.

Visualization of Embeddings. MRLR framework can generate meaningful embeddings that help interpret recommendation results. To show this, we visualize the embeddings

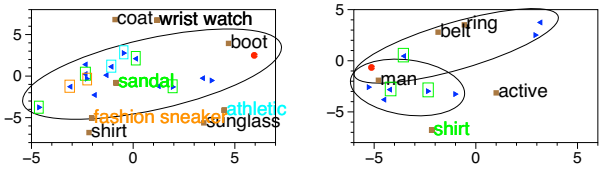

Figure 3: Visualization of user (red dot), item (blue triangle), and category (brown square) embeddings in a two dimensional space. Left-pointing triangles are rated items; right-pointing triangles are recommended items. The category of an item is labelled by a rectangle whose color is the same as its belonging category.

of users, items and categories learnt by MRLR in a two dimensional space using t-SNE [Maaten and Hinton, 2008]. Figure 3 illustrates the results of two examples in the Clothing dataset. For conciseness, we do not visualize the other datasets, however, similar observations as below can be obtained: 1) the rated items and the recommended items are generally clustered. This indicates certain similarity among the rated items and the recommended items to the same user. 2) each cluster is located at the side of the user, and the user is represented as an endpoint of these clusters, indicating that user preference can be manifested as the direction along which the rated items are clustered. This suggests that the recommendations are determined by both rated items and user preferences. Finally, we note that the categories of recommended items are overlapped with those of the rated items. For instance, for the user in the right plot the overlapped category is Shirts, indicating user preference over shirts. For the user in the left plot the overlapped categories are Athletic, Fashion Sneakers, and Sandals, indicating that the user has a more diverse set of preferences. These observations show that MRLR can capture meaningful item relationships in multiple levels of item organizations – individual items, items in the same category, and items rated by the same user.

4.3 Comparative Results

Table 2 summarizes the performance of all comparison methods. Two views are considered: 'All Users' indicates all users are considered in the test data; while 'Cold Start' indicates only users with less than 5 ratings are involved in the test data. Several interesting findings are observed as follows.

Among the latent factor model based methods (PMF, BPR and FM), PMF performs the worst, as it is the basic rating prediction method without considering any auxiliary information. FM incorporates item category as auxiliary input, significantly outperforms PMF, indicating the effectiveness of item category for better recommendation. Interestingly, the performance of FM is worse than that of BPR. This verifies that personalized ranking is more effective than rating prediction in real-world recommendation scenarios.

The RL methods, including Item2Vec, MetaProd2Vec, CoFactor and User2Vec, generally perform better than latent factor based methods, despite being rating prediction models. This confirms that representation learning is more effective than latent factor models for recommendation. Among them, Item2Vec performs worse than MetaProd2Vec. This observation further confirms the previous conclusion that item cate-

Table 2: Performance (AUC) of comparison methods, where the best performance is highlighted in bold; the second best performance of other methods is marked by '*'; 'Improve' indicates the improvements of MRLR relative to the '*' results.

Datasets	Cases	PMF	BPR	FM	Item2Vec	MP2Vec	CoFactor	User2Vec	MRLR	Improve
Clothing	All Users	0.5255	0.6151	0.5972	0.6429	0.6600*	0.6012	0.6249	**0.7058**	6.94%
	Cold Start	0.5291	0.6135	0.5969	0.6426	0.6602*	0.5984	0.6203	**0.7022**	6.36%
Electronics	All Users	0.6595	0.7178	0.7066	0.7529	0.7604*	0.7000	0.7121	**0.7932**	4.31%
	Cold Start	0.6558	0.7161	0.7010	0.7535	0.7631*	0.6937	0.7107	**0.7935**	3.98%
Sports	All Users	0.6136	0.6992	0.6856	0.7015	0.7148*	0.6693	0.6852	**0.7569**	5.89%
	Cold Start	0.6175	0.7013	0.6861	0.7063	0.7149*	0.6679	0.6883	**0.7541**	5.48%
Home	All Users	0.6319	0.6930	0.6795	0.7297	0.7455*	0.6737	0.6969	**0.7772**	4.25%
	Cold Start	0.6408	0.6911	0.6841	0.7317	0.7449*	0.6731	0.6917	**0.7763**	4.22%

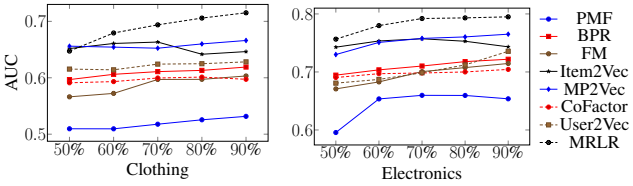

Figure 4: Impacts of data sparsity on the performance.

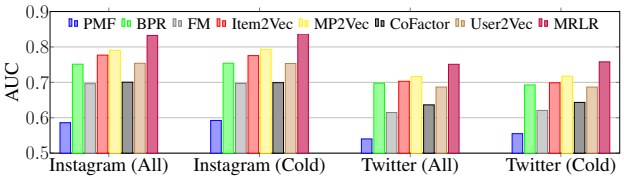

Figure 5: Comparative results on Instagram and Twitter.

gory is useful to improve recommendation performance.

CoFactor and User2Vec consider personalization in addition to item embedding. CoFactor is equivalent to the CMF method as it simultaneously factorizes user-item and item-item co-occurrence matrices with shared item latent factors, while User2Vec adopts CBOW to integrate personalization. Theoretically, the performance of the two methods should be better than that of Item2Vec, since they can provide users with personalized item list. We empirically find that User2Vec outperforms CoFactor, but both are slightly worse than Item2Vec. However, our proposed variant RL with Skip-gram outperforms Item2Vec, by 6.37% on average (Figure 1). Hence, we conjecture that considering personalization with Item2Vec helps improve recommendation performance, but CMF, CBOW are less effective than Skip-gram in incorporating item co-occurrence relationships with personalization.

Overall, compared with all the other methods, MRLR performs the best by learning user and item embeddings from a multi-level item organization, i.e., items in user-specific ranked list, items in the same category, and individual items. The improvements w.r.t. 'All User' and 'Cold Start' are 5.35%, 5.01% on average (both with p-value $< .01$), respectively. This implies that recommendation performance can be further enhanced by appropriately considering multi-level representation learning and personalized ranking.

Impacts of Data Sparsity. We further study the impacts of data sparsity on the recommendation performance. Figure 4 depicts the variation of performance of all methods on Clothing & Electronics when the percentage of training data size w.r.t. the overall data size increases from 50% to 90%. We observe that MRLR consistently outperforms other methods across all levels of data sparsity. Furthermore, the performance of MRLR with data sparsity at 60% is better than that of any of other methods with data sparsity at 90%. Such observations also hold in other datasets, showing that MRLR can achieve better performance even with high data sparsity.

Generalizability. To evaluate the generalizability of MRLR, we further collect data of Foursqure check-in performed over 3 weeks in 4 European capital cities (Amsterdam, London, Paris, Rome), published on Instagram (31,872 users perform 198,801 check-in at 41,387 locations that belong to 492 categories) and Twitter (18,522 users; 109,790 check-in; 38,855 locations; 482 categories). Figure 5 compares the performance of MRLR and the other methods. As in the previous setting, MRLR significantly outperforms (p-value $< .01$) the second best method MetaProd2Vec by 5.10% on 'All Users' and 5.62% on 'Cold Start'. These results show that MRLR can be effective in multiple recommendation tasks.

5 Conclusions

Representation learning (RL) has drawn much attention in recommendation, due to its effectiveness in capturing local item relationships. However, all existing RL based methods model recommendation as a rating prediction problem while recommendation is essentially a personalized ranking one. Besides, they all neglect multi-level organizations of items for fine-grained item relationships. Hence, this paper proposes a multi-level RL framework for personalized ranking – MRLR, which learns user and item embeddings from a multi-level item organization for better recommendation. MRLR, therefore, benefits from RL as well as achieves the goal of personalized ranking. Empirical validation on real-world datasets shows that MRLR significantly outperforms state-of-the-art algorithms.

Acknowledgements

This work is supported by the SIMTech-NTU Joint Laboratory on Complex Systems. This work is partially funded by the Social Urban Data Lab (SUDL) of the Amsterdam Institute for Advanced Metropolitan Solutions (AMS).

References

[Barkan and Koenigstein, 2016] Oren Barkan and Noam Koenigstein. Item2vec: Neural item embedding for collaborative filtering. *IEEE Workshop on MLSP*, 2016.

[Bengio *et al.*, 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *JMLR*, 3(Feb):1137–1155, 2003.

[Chen *et al.*, 2012] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. Svdfeature: a toolkit for feature-based collaborative filtering. *JMLR*, 13(1):3619–3622, 2012.

[Covington *et al.*, 2016] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *RecSys*, pages 191–198. ACM, 2016.

[Grbovic *et al.*, 2015] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. E-commerce in your inbox: Product recommendations at scale. In *KDD*, pages 1809–1818. ACM, 2015.

[He *et al.*, 2016] Ruining He, Chunbin Lin, Jianguo Wang, and Julian McAuley. Sherlock: sparse hierarchical embeddings for visually-aware one-class collaborative filtering. In *IJCAI*, pages 3740–3746, 2016.

[Karatzoglou *et al.*, 2010] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *RecSys*, pages 79–86. ACM, 2010.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434. ACM, 2008.

[Le and Mikolov, 2014] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196, 2014.

[Lee and Seung, 2001] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2001.

[Levy and Goldberg, 2014] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *NIPS*, pages 2177–2185, 2014.

[Liang *et al.*, 2016] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M Blei. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *RecSys*, pages 59–66. ACM, 2016.

[Maaten and Hinton, 2008] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(Nov):2579–2605, 2008.

[McAuley *et al.*, 2015] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52. ACM, 2015.

[Mikolov *et al.*, 2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[Mikolov *et al.*, 2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[Mnih and Salakhutdinov, 2007] Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2007.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43, 2014.

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

[Rendle, 2010] Steffen Rendle. Factorization machines. In *ICDM*, pages 995–1000. IEEE, 2010.

[Shi *et al.*, 2012] 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*, pages 139–146. ACM, 2012.

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

[Singh and Gordon, 2008] Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *KDD*, pages 650–658. ACM, 2008.

[Steck, 2013] Harald Steck. Evaluation of recommendations: rating-prediction and ranking. In *RecSys*, pages 213–220. ACM, 2013.

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

[Vasile *et al.*, 2016] Flavian Vasile, Elena Smirnova, and Alexis Conneau. Meta-prod2vec: Product embeddings using side-information for recommendation. In *RecSys*, pages 225–232. ACM, 2016.

[Weimer *et al.*, 2007] Markus Weimer, Alexandros Karatzoglou, Quoc Viet Le, and Alex Smola. Maximum margin matrix factorization for collaborative ranking. In *NIPS*, pages 1–8, 2007.

[Yang *et al.*, 2016] Jie Yang, Zhu Sun, Alessandro Bozzon, and Jie Zhang. Learning hierarchical feature influence for recommendation by recursive regularization. In *RecSys*, pages 51–58. ACM, 2016.