Rank and Rate: Multi-task Learning for Recommender Systems

Guy Hadash
IBM Research
Haifa, Israel
guyh@il.ibm.com

Oren Sar Shalom
Intuit
Hod HaSharon, Israel
oren.sarshalom@gmail.com

Rita Osadchy
University of Haifa
Haifa, Israel
rita@cs.haifa.ac.il

ABSTRACT

The two main tasks in the Recommender Systems domain are the ranking and rating prediction tasks. The rating prediction task aims at predicting to what extent a user would like any given item, which would enable to recommend the items with the highest predicted scores. The ranking task on the other hand directly aims at recommending the most valuable items for the user. Several previous approaches proposed learning user and item representations to optimize both tasks simultaneously in a multi-task framework. In this work we propose a novel multi-task framework that exploits the fact that a user does a two-phase decision process - first decides to interact with an item (ranking task) and only afterward to rate it (rating prediction task).

We evaluated our framework on two benchmark datasets, on two different configurations and showed its superiority over state-of-the-art methods.

KEYWORDS

Recommender Systems; Collaborative Filtering

ACM Reference Format:
Guy Hadash, Oren Sar Shalom, and Rita Osadchy. 2018. Rank and Rate: Multi-task Learning for Recommender Systems. In *Twelfth ACM Conference on Recommender Systems (RecSys '18), October 2–7, 2018, Vancouver, BC, Canada.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3240323.3240406

1 INTRODUCTION

With the flourishing amount of products and information available on the web, recommender systems have become a prominent and useful tool. The benefits of personalized recommendations are unquestionable, hence the growing attention in this research topic. There are two main approaches for generating recommendations using Collaborative Filtering (CF) methods: rating prediction task and ranking task. At first, the dominant approach was the former. This task, encouraged by the Netflix Prize [1], aims at predicting the rating a user would assign to given items. Given the predicted ratings that a user would assign to all items in the catalog, the recommended list can be composed of the items with the highest expected ratings. However, this approach introduces two major disadvantages:

(1) It does not prioritize the head of the recommended list [3]. Since the final objective is to suggest interesting and valuable items for users, a small prediction error in the low rated items is not equivalent to a small prediction error in the high rated items.
(2) This task can be done only on rated items. As such it requires logged data of explicit ratings, which is much sparser than implicit feedback. Additionally, it suffers from selection bias, as user's decision not to interact with certain items is ignored [12].

Therefore, in recent years more focus is given to the ranking task (also known as the selection task or the top-k recommendation problem), which is considered to better reflect user's needs [5]. The goal of the ranking task is to compute rank scores, which may not predict the assigned rating, but rather they are directly used to generate the list of recommended items.

Although the ranking task could be considered to be more important than the rating task, each of them encodes different bits of information and they may complete each other. Therefore, excelling in the secondary task may improve users and items representations and to lead to better performance on the primary task. This intuition paved the way for several frameworks that perform multi-task learning [9–11, 21].

In this paper, we refine the aforementioned intuition and argue that these two problems are not completely separate, but are part of a single process. A user first decides to interact with an item and subsequently decides on the explicit rating. We therefore design a multi-task learning approach that captures this two-phase decision process, and present its improved performance.

The rest of the paper is organized as follows. We start by giving the background (Section 2) needed for problem formulation. We then review the related works in Section 3. We present our novel approach in Section 4. Section 5 includes the evaluation of the proposed method and the empirical comparison with the related work. Finally, Section 6 concludes the paper.

2 BACKGROUND

Collaborative Filtering (CF) methods receive as input a usage matrix D consisting of N users and M items. Matrix D may contain explicit ratings r_{ui} that accompany the historical interactions.

Latent factor models for CF represent each user u and item i by some low dimensional vectors p_u and q_i correspondingly. The predicted score of a given user-item pair (u, i) is determined by: $s_{ui} = p_u^T \cdot q_i$. This score can be used for either ranking or rating prediction. Ranking algorithms usually learn user bias terms b_u, which are added to the predicted scores. Rating prediction algorithms may add to the predicted score the user and item biases, alongside with the mean rating.

RecSys '18, October 2–7, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5901-6/18/10…$15.00
https://doi.org/10.1145/3240323.3240406

The various algorithms differ in their objective function and means to generate the internal representation for users and items. Such models for the rating prediction task are SVD[15] and SVD++[7]; Cornerstone ranking oriented algorithms under this paradigm are BPR [16], WRMF[6], CDAE[23].

Multi-task learning (MTL) [2] deals with problems where multiple tasks with some commonalities are given. The learning procedure optimizes these tasks simultaneously, using representation sharing. Due to the common traits of the tasks, such an optimization method can improve generalization and accuracy for the task-specific models, when compared to training the models separately.

3 RELATED WORK

MTL has been shown to be beneficial in variety of tasks, including the problem of generating personalized recommendations, that we focus here. However, the research on this problem is still sparse. The first work in this area ([21]) suggested training simultaneously a ranking and a rating prediction algorithms with shared embedding matrices. The ranking algorithm was chosen to be ListRank [20] and the rating algorithm was Probabilistic Matrix Factorization [13]. Later, the authors of [11] proposed to use CLiMF [19] as the ranking algorithm and reported improved results. These works are conceptually similar, as they differ only in the choice of the concrete selection of the rating prediction algorithm. One can unify these works in a framework that uses the *same* representation for both learning tasks, i.e., the same user and item vectors are used for ranking the items and for predicting the actual ratings.

Another line of work incorporates both implicit feedback and explicit ratings for either ranking or rating prediction tasks. A landmark algorithm that falls under this umbrella is SVD++[7], which solves the rating prediction task. The user representation generated by this algorithm considers all items for which the user made an implicit feedback. It further considers all explicit ratings also as implicit feedback interactions.

On the other side, several ranking-oriented algorithms were proposed, that integrate explicit ratings. For instance, BPR [16] is a seminal ranking algorithm. The algorithms presented in [14, 22] enhance it by allowing it to support explicit ratings. For last, the work presented in [9] integrates the rating prediction algorithm SVD++ with the ranking algorithm xCLiMF[18]. These types of work, however, do not serve as a general framework that unifies ranking and rating prediction tasks. Instead they are tailored to specific algorithms and generalizing them to any other algorithm may not be trivial.

4 OUR APPROACH

In this work, we argue that the ranking and rating prediction tasks are not completely separate problems, but are part of a single process. We accompany the description of our intuition with examples from the movies domain, although the presented concepts are general and can be applied to other domains as well. First, the user decides to interact with a specific item (watch a movie). The ranking task, which is also known as the selection task, aims at forecasting this choice of the user. The decision made by the user is based on several weighted criteria. For example, the user may decide to watch only Hollywoodian movies and highly prefers comedies. After interacting with the item, in post-consumption perspective, the user becomes more familiar with the item's traits, may notice nuances and her perception of the item is modified. In our running example, the user may understand that the movie is funnier than expected. Then the user may leave an explicit rating that summarizes the assessment of the user toward the item. At this point, the user's system of considerations is different comparing to the previous choice. For instance, given that the user has watched the movie, the assigned rating may be heavily influenced by the plot, and less affected by whether the movie was shot in Hollywood. The rating prediction task is aligned with this decision.

We therefore design a general framework that unifies the ranking and rating prediction tasks while supporting the described two-phase decision process of ranking and rating. We hereinafter dub the proposed algorithm as "Rank and Rate", or RnR for short. Let $(\mathcal{R}, \mathcal{P})$ be a pair of any latent factor models that solve the ranking and the rating prediction tasks correspondingly. These algorithms will serve as the underlaying algorithms for RnR. Our proposed approach is a general framework, and is agnostic to the concrete choice of the underlaying algorithms, as long as they can be optimized by Stochastic Gradient Descent based optimization methods. They can even vary in the input, required at training time, or to follow different optimization approaches (i.e., point-wise, pair-wise or list-wise). For instance, a possible choice of ranking algorithm can be WRMF[6], which requires only the identities u, i of the target user and item. Another natural candidate for the ranking algorithm is BPR[16], which in a addition to the user and item identities, requires a negative sampled item. For the rating prediction algorithm one can consider SVD[8] or SVD++[7]. While the former requires only the identities u, i and the explicit rating, the latter also requires the set of all implicit feedbacks given by the user u.

Let $L_P(\mathcal{P}, D; U, I)$ be the loss achieved by algorithm $\mathcal{P}$ on dataset D with user and item embedding matrices U and I[1]. Similarly, we define the loss of the ranking algorithm $\mathcal{R}$ as $L_R(\mathcal{R}, D; U, I)$. RnR performs multi-task learning, and optimizes the parameter sets of $\mathcal{R}$ and $\mathcal{P}$ simultaneously, while sharing the embedding matrices U and I. Unlike previous methods, our approach applies the two-phase decision process by modifying the user and item representations for the *rating prediction task*. Namely, we iterate over the dataset D and simultaneously optimize both $\mathcal{R}$ and $\mathcal{P}$. While $\mathcal{R}$ uses the raw embedding matrices U and I, $\mathcal{P}$ uses parameters that support the second decision made by the user. Specifically, we learn an item deviation matrix I^d which models a post-consumption perspective of the users toward items. For each item i, it gives us a deviation item vector q_i^d, which we add to the original item vector q_i to obtain the modified item vector q_i^{post}. Therefore, q_i^{post} reflects the post-consumption perception of the item. We perform an unweighted sum $q_i^{post} = q_i + q_i^d$, since the parameters in I^d can learn the relative importance of each component in the sum. Next, we map the user and the updated item vectors to a new feature space which reflects the post-consumption decision process of the user. We implement the mapping by a fully connected layer (FC) with a non-linear activation function. We tie the parameters of the user and item

[1]We assume a single item embedding matrix, but it is straightforward to support multiple matrices

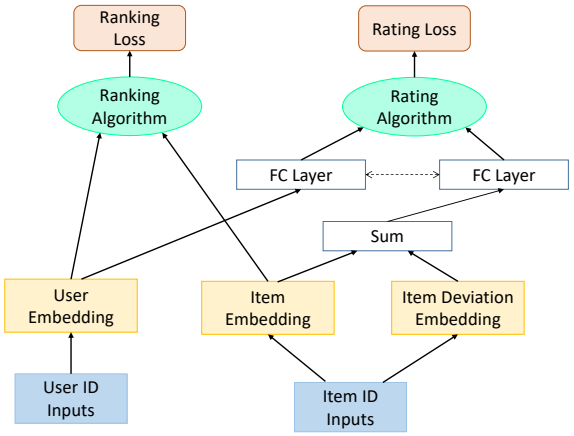

Figure 1: Model Architecture - illustrates the two-phase process, where we first rank using the raw embeddings, and then rate using the transformed embeddings.

FC layers, hence the new representations p_u^P and q_i^P of users and items for the rating prediction task are projected to the same latent space. Formally, $p_u^P = FC_\theta(p_u), q_i^P = FC_\theta(q_i^{post})$, where $FC_\theta(\cdot)$ denotes the output of a fully connected layer parameterized by θ. The architecture of RnR is shown in Figure 1.

Our proposed framework mimics the decisions making process as follows. We go over all tuples $(u, i, r) \in D$. For each, we first invoke $\mathcal{R}$ to predict the ranking choice. We then invoke $\mathcal{P}$ with the modified parameters as explained above, to predict the rating choice. We define the objective function as follows,

$$\begin{aligned} O = \min_{U,I,I^d,\theta} \; & \alpha \cdot L_R(\mathcal{R}, D; U, I) \\ & + (1-\alpha) \cdot L_P(\mathcal{P}, D; FC_\theta(U), FC_\theta(I + I^d)) \\ & + \lambda(\|U\|^2 + \|I\|^2 + \|I^d\|^2 + \|\theta\|^2) \end{aligned} \tag{1}$$

where α and λ are hyper-parameters that control the relative contribution of each individual task and the regularization factor, respectively.

5 RESULTS

In our experiments, we test the performance of proposed RnR method, which combines the multi-task learning of ranking and rating prediction tasks with the two-phase decision process. We compare it to two types of baselines: 1) a vanilla multi-task framework that learns these two tasks simultaneously, but without the two-phase decision process[11, 21] and 2) single-task algorithms: Collaborative Denoising Auto-Encoders[23], Bayesian Personalized Ranking[16], and SVD[15]. We also report the results of Popularity, which is always predicting the top k most frequent items in train, for completeness.

For the vanilla multi-task learning, given the underlaying algorithms for ranking and rating prediction, $\mathcal{R}$ and $\mathcal{P}$, we optimizes the parameters using the combined objective:

$$\begin{aligned} O = \min_{U,I} \; & \alpha \cdot L_R(\mathcal{R}, D; U, I) + (1-\alpha) \cdot L_P(\mathcal{P}, D; U, I) \\ & + \lambda(\|U\|^2 + \|I\|^2). \end{aligned} \tag{2}$$

As in RnR, the role of α is to control the relative contribution of each individual task.

For ranking, we choose two different algorithms: Collaborative Denoising Auto-Encoders[23] and Bayesian Personalized Ranking[16]. For rating prediction we use SVD[15] (the algorithms are detailed below).

5.1 Underlaying Algorithms

Collaborative Denoising Auto-Encoders (CDAE) [23] is an encoder-decoder architecture, that achieves state-of-the-art results on the personal recommendations task. It is consists of three main concepts. First, is the *Auto-Encoder*, which is given as input a set of items a user has interacted with. Then it generates an internal representation of the user (the so-called user vector), which in turn allows to reconstruct the original input. The second concept, *Collaborative*, means that the identity of the user is also given as input, which allows to refine the internal representation of the user. The last concept *Denoising* improves the generalization of the model by asking to reconstruct the complete historical usage from a partially observed list of items.

Bayesian Personalized Ranking (BPR) method [16] is a simple, yet effective and widely adopted implicit feedback MF method. BPR is optimized for correctly ranking observed user-item interactions over non-observed ones.

SVD[15] is a seminal rating prediction algorithm. It factorizes a usage matrix (with explicit feedback) into two low rank matrices, for users and items. This distributed representation, together with learned user and item biases, is learned to minimize the squared error of the reconstructed ratings.

5.2 Datasets

Our evaluation is done on two real-world datasets, from different domains.

MovieLens[2]: This dataset contains about 1 million ratings that were applied to more than 3,700 movies by more than 6,000 users.

Yelp:[3] This is the most recent dump of the Yelp challenge. It contains about 3.5M reviews on about 170,000 businesses made by about 295,000 users.

Each user interaction in these datasets is associated with an explicit rating on a 1-5 Likert. Users with less than 4 interactions were omitted. Datasets statistics are summarized in Table 1.

5.3 Evaluation Protocol and Experiments

Using each dataset, we evaluated our approach and the baselines using an "All But (Last) One" protocol (i.e., the hidden item belongs to the chronologically last interaction), namely, we remove the last rated item for n users from the training set, and put half of the removed items in the validation set and the other half in the test set. We used $n = 5,000$ in the MovieLens experiments and $n = 50,000$

[2]http://grouplens.org/datasets/movielens/1m/
[3]http://www.yelp.com/dataset/challenge

Dataset	Users	Items	Ratings	Spareness	Ave. Ratings/User
MovieLens1M	6,040	3,706	1,000,209	95.53%	165.60
Yelp 2018	294,608	169,603	3,785,528	99.99%	12.849

Table 1: Statistics of datasets used in the experiments.

	MovieLens 1M		Yelp 2018	
Algorithm	Accuracy	MRR	Accuracy	MRR
Popularity	0.0278	0.0035	0.0166	0.0053
SVD	0.0673	0.0120	0.0043	0.0010
CDAE	0.0926	0.0146	0.0305	0.0107
Vanilla(CDAE, SVD)	0.1040	0.0189	0.0318	0.0115
RnR(CDAE, SVD)	**0.1236**	**0.0497**	**0.0400**	**0.0137**
BPR	0.0659	0.0222	0.0196	0.0071
Vanilla(BPR, SVD)	0.0732	0.0244	0.0207	0.0054
RnR(BPR, SVD)	**0.0752**	**0.0263**	**0.0238**	**0.0077**

Table 2: Experiments results, higher is better.

in Yelp experiments. The results are reported on Recall@k (denoted by Accuracy@k) and MRR@k [17], for k=10.

We implemented our and baseline methods using TensorFlow [4] employed with the AdaGrad optimizer [4], with learning rate of 0.001. User and item embedding size was fixed to be 50. We did modest grid search over $\alpha \in [0.9, 0.95, 1.0]$ and regularization $\lambda \in [0.01, 0.001]$.

We initialized the deviation matrix I^d and biases with zeros, since its used in summation with other embeddings or scalars. All other parameters were initialized using Xavier initialization.

5.4 Results

The complete list of results is reported in Table 2. We first observe that CDAE outperforms BPR, and in general BPR achieves better results than SVD. Second, combining ranking and rating algorithms using the vanilla approach always improves over single-task algorithms. We can further note that our method significantly improves the vanilla approach. This is attributed to the support in the two-phase decision process made by users.

6 CONCLUSIONS

In this work, we showed a generic, yet efficient way of improving a recommendation system which combines multi-task learning of ranking and rating prediction with two-phase decision process made by users. For the latter, we used an explicit rating information, which is available in many cases. We showed that the proposed framework can combine different underlying algorithms, improving them significantly in all cases. Hence, a practitioner can use the proposed framework with any other ranking and raring predictions methods.

We expect that the results of this work will inspire further research on using the explicit signal in efficient ways.

[4]https://www.tensorflow.org/

REFERENCES

[1] James Bennett, Stan Lanning, and others. 2007. The netflix prize. In *Proceedings of KDD cup and workshop*, Vol. 2007. New York, NY, USA, 35.

[2] Rich Caruana. 1998. Multitask learning. In *Learning to learn*. Springer, 95–133.

[3] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 39–46.

[4] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[5] Asela Gunawardana and Guy Shani. 2009. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research* 10, Dec (2009), 2935–2962.

[6] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 263–272.

[7] Yehuda Koren. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*. ACM, New York, NY, USA, 426–434. DOI: http://dx.doi.org/10.1145/1401890.1401944

[8] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[9] Gai Li and Qiang Chen. 2016. Exploiting explicit and implicit feedback for personalized ranking. *Mathematical Problems in Engineering* 2016 (2016).

[10] Gai Li, Chao-bo He, Liyang Wang, Jin-cai Pan, Qiang Chen, and Lei Li. 2016. An Unified One Class Collaborative Filtering Algorithm. In *Chinese National Conference on Social Media Processing*. Springer, 267–273.

[11] Gai Li, Zhiqiang Zhang, Liyang Wang, Qiang Chen, and Jincai Pan. 2017. One-class collaborative filtering based on rating prediction and ranking prediction. *Knowledge-Based Systems* 124 (2017), 46–54.

[12] Benjamin M Marlin and Richard S Zemel. 2009. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the third ACM conference on Recommender systems*. ACM, 5–12.

[13] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[14] Weike Pan, Hao Zhong, Congfu Xu, and Zhong Ming. 2015. Adaptive Bayesian personalized ranking for heterogeneous implicit feedbacks. *Knowledge-Based Systems* 73 (2015), 173–180.

[15] Arkadiusz Paterek. 2007. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, Vol. 2007. 5–8.

[16] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

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

[18] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, and Alan Hanjalic. 2013. xCLiMF: optimizing expected reciprocal rank for data with multiple levels of relevance. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 431–434.

[19] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. 2013. CLiMF: Collaborative Less-Is-More Filtering.. In *IJCAI*. 3077–3081.

[20] Yue Shi, Martha Larson, and Alan Hanjalic. 2010. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 269–272.

[21] Yue Shi, Martha Larson, and Alan Hanjalic. 2013. Unifying rating-oriented and ranking-oriented collaborative filtering for improved recommendation. *Information Sciences* 229 (2013), 29–39.

[22] Sheng Wang, Xiaobo Zhou, Ziqi Wang, and Ming Zhang. 2012. Please spread: recommending tweets for retweeting with implicit feedback. In *Proceedings of the 2012 workshop on Data-driven user behavioral modelling and mining from social media*. ACM, 19–22.

[23] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *WSDM*.