Abstract—Recommendation methods based on deep learning frameworks have drastically increased over recent years, covering virtually all the sub-topics in recommender systems. Among these topics, one-class collaborative filtering (OCCF) as a fundamental problem has been studied most extensively. However, most of existing deep learning-based OCCF methods are essentially focused on either defining new prediction rules by replacing conventional shallow and linear inner products with a variety of neural architectures, or learning more expressive user and item factors with neural networks, which may still suffer from the inferior recommendation performance due to the underlying preference assumptions typically defined on single items. In this paper, we propose to address the limitation and justify the capacity of deep learning-based recommendation methods by adapting the setwise preference to the underlying assumption during the model learning process. Specifically, we propose a new setwise preference assumption under the neural recommendation frameworks and devise a general solution named DeepSet, which aims to enhance the learning abilities of neural collaborative filtering methods by activating the setwise preference at different neural layers, namely 1) the feature input layer, 2) the feature output layer, and 3) the prediction layer. Extensive experiments on four commonly used datasets show that our solution can effectively boost the performance of existing deep learning-based methods without introducing any new model parameters.

Index Terms—One-Class Collaborative Filtering, Deep Learning, Setwise Preference

I. INTRODUCTION

The explosive growth of web resources intensifies the need to precisely match users with specific information, making recommender systems increasingly important. As one of the building blocks in recommendation methods, collaborative filtering has been extensively studied for two fundamental recommendation tasks, i.e., rating prediction and item ranking. When the input data consists of only one-class (user, item) interactions such as views, clicks or purchases, ranking-oriented collaborative filtering is also referred as one-class collaborative filtering (OCCF) [1].

Due to the one-class characteristic that makes it hard to learn effective models to distinguish items preferred by users from un-interesting ones, numerous methods have been proposed to address the problem from different perspectives. The very first two works [1], [2] propose to fit the one-class data with matrix factorization using a square loss, where the observed interactions are assumed to be positive feedback while the unobserved (user, item) pairs are sampled or weighted to be negative feedback. Under such a matrix factorization scheme, the prediction score that quantifies how much a user would like to interact with an item is usually calculated as an inner product between the user-specific latent vector and the item-specific latent vector, where the rule or formula used to calculate the preference score for a (user, item) pair is referred as the prediction rule. Some previous works [3] focus on refining the prediction rules by introducing new model parameters, which often increase the model complexity. However, some empirical studies show that it is more effective to enhance the recommendation performance by refining the learning patterns with more appropriate assumptions, which define the ways how (user, item) pairs are associated in loss functions, i.e., preference assumptions. BPR (Bayesian personalized ranking) [4] is a representative work that refines the typical pointwise assumption on single items, which learns its model based on pairwise comparisons between an observed pair and an unobserved pair. Under the pairwise learning framework of BPR, GBPR [5] and CoFiSet [6] further extend the individual preference in BPR to group preference and setwise preference. Both of them achieve very promising results. It is worthy to point out that alternatives on preference assumptions usually would not change the prediction rule.

Inspired by the great success of deep learning in various domains, a number of deep learning architectures have been dramatically adapted to one-class collaborative filtering [7], [8]. There are mainly two ways to “deep learn” traditional OCCF methods. One is using neural networks to model the preference scores for (user, item) interactions, e.g., NeuMF [7] can be viewed as deep learning extensions of LogisticMF [9] by replacing the original prediction rules with multilayer networks. The other way is to use neural networks to yield more expressive user and item features [8], [10].

Even though these works claim that they achieve the state-of-the-art performance superior to traditional methods, some recent works [11], [12] challenge the so-claimed progress. We also have similar findings, i.e., most of these deep learning-based recommendation methods (without pre-training) indeed suffer from inferior performance compared with traditional non-neural methods in our empirical studies. Is it true that deep learning for recommendation just appear better than

*: Corresponding author.
it is? We believe that these deep learning techniques have sufficient capacity to boost the state-of-the-art recommendation performance, and the key is how to fully encourage their superiority in capturing complex patterns. However, as we discussed above, most of existing neural recommendation methods are essentially defining new prediction rules or just neutralizing the typical user and item factors by stacking nonlinear neural computations, which may not guarantee better performance and increase the model complexity instead. One obvious limitation of these methods is that they adopt classical preference assumptions on single items or item pairs. In other words, they learn all model parameters by treating one (user ID, item ID) pair or one (user ID, observed item ID, unobserved item ID) triple as an individual training instance, which may not afford to provide enough information for so many parameters to learn effectively.

Motivated by the significant performance improvements of CoFiSet [6], we employ the setwise preference into neural model learning process, aiming to facilitate the real potential of neural networks on handling complex patterns and further enhance the existing deep learning-based recommendation methods. Specifically, the contributions of our work are as follows.

- We adapt the setwise preference to neural recommendation models and devise a novel general solution for deep recommendation with MLP to model setwise preference, i.e., DeepSet, which refines the individual preference assumptions of the typical deep learning-based recommendation methods. To the best of our knowledge, it is the first attempt to introduce setwise preference into deep learning-based recommendation methods for the studied problem.
- We develop three variants to incorporate this novel assumption by activating the setwise preference at different neural layers, namely the feature input layer, the feature output layer, and the prediction layer, corresponding to DeepSet(f), DeepSet(fo) and DeepSet(p), respectively. Notice that our solution does not introduce any additional parameters.
- Note that our proposed DeepSet is vertical to existing deep recommendation models based on or integrated with MLP, which means that DeepSet can be instantiated with various MLP-based recommendation methods. In this paper, we take two very classical neural methods GMF and NeuMF as base models, instantiating DeepSet to derive new models.
- We conduct extensive experiments on four public datasets, which demonstrate the effectiveness of our solution over the neural collaborative filtering framework.

II. RELATED WORK

Earlier literatures mainly design OCCF models from two perspectives, i.e., the prediction rule and the model learning pattern under some specific preference assumptions. With the commonly used prediction rule that computes the inner product, the typical learning pattern is to take the observed items as positive feedback and sample some unobserved items as negative feedback, which usually transforms the one-class problem into a classification task under some preference assumptions on items. LogisticMF [9] is a representative work that makes recommendation based on the basic prediction rule with a logistic loss. With this basic framework, many works [3] focus on refining the prediction rule, which in many cases may not achieve significant improvements.

Unlike typical methods under the pointwise learning scheme, BPR [4] is the first work that attempts to design new models by refining the learning patterns, which directly optimizes the ranking-oriented metric AUC and achieves excellent performance in many cases. Many follow-up works focus on enhancing BPR by introducing more pairwise comparisons [13] and additional information [14], which usually involves more parameters. But there are several works enhancing BPR greatly from the perspective of preference assumptions. Considering that the individual and independent preference assumption in BPR may not always be true, GBPR [5] proposes group preference defined on a group of users to facilitate better parameter learning. Similarly, instead of relying on pairwise comparison between two items, CoFiSet [6] refines the assumption in BPR by extending the comparisons between two items to two item-sets. Different from the widely used pointwise and pairwise learning paradigms, listwise methods [15] try to directly learn a ranking for a list of items consisting of observed items and unobserved items, but many of these methods suffer from the rather poor efficiency greatly. The setwise preference has also been employed with listwise learning to show competitive performance [16].

With the revolutionary expansion of deep learning, many traditional recommendation methods have been extended to deep learning versions. NeuMF [7] is a typical neural recommendation model that defines a new prediction rule by integrating a classical MF with a multi-layer perceptron to replace the inner product of LogisticMF. In the same way, NeuPR [17] extends BPR under the framework of NeuMF by computing the vector difference at the feature input layer. Likewise, CDAE [18] can be considered as a neural version of the classical similarity-based method [3] using autoencoder. There are also some works using neural networks to learn more sophisticated user/item features. DeepCF [10] learns user/item vectors with MLP as the final embeddings for further computation. It can be seen that these neural recommendation methods are using networks as a replacement for traditional inner product or single-layer user/item vector, which actually suffer from inferior performance in most cases as [11], [12] point out. We can hardly find any deep learning based recommendation methods that refine classical models with new learning patterns under some specific preference assumptions, except [19], [20], which enhance the recommendation performance by employing some reinforcement learning techniques.

III. BACKGROUND

In this section, we first present the problem definition, and then introduce GMF and NeuMF [7] in terms of its prefer-
ence assumption, neural architecture (i.e., prediction rule) and learning objectives, since it serves as the basic model which we instantiate our solution with.

A. Problem Definition

Suppose there are $n$ users and $m$ items in a recommender system, which are denoted by $\mathcal{U}$ and $\mathcal{I}$, respectively. The observed items w.r.t. each user $u \in \mathcal{U}$ are provided, denoted by $\mathcal{I}_u = \{i | (u, i) \in \mathcal{R}^+\}$, which means user $u$ has interacted with these items. Our task is to generate a ranking list of items to be potentially interacted with by a user $u$ from the unobserved items $\mathcal{I}\backslash \mathcal{I}_u$. This problem is referred as one-class collaborative filtering (OCCF).

B. Preference Assumption in NeuMF

Same as LogisticMF [9], NeuMF [7] adopts the preference assumption defined on single items under the pointwise learning scheme, which can be formulated as follows:

$$r_{ui} = 1, r_{uj} = 0, \; i \in \mathcal{I}_u, \; j \in \mathcal{I}\backslash \mathcal{I}_u,$$

where $r_{ui}$ denotes the pre-defined preference score of user $u$ on an observed item $i$, and $r_{uj}$ denotes the pre-defined preference towards an unobserved item $j$. We can see that NeuMF treats each (user, item) interaction as an individual sample for model learning.

C. Architecture of NeuMF

NeuMF [7] comprises a shallow network extended on inner product and a deep network constructed by multiple neural layers. Horizontally, NeuMF contains two components, i.e., generalized matrix factorization (GMF) and multi-layer perceptron (MLP). GMF first computes the element-wise product of a user’s vector $U^G_u$ and an item’s vector $V^G_i$, then multiplies the resulting vector with a weight vector $W_G$ to yield the final prediction. For user $u$ and item $i$, the predicted preference score $\hat{r}_{ui}$ with GMF can be calculated as follows:

$$\hat{r}_{ui}^G = \sigma((U^G_u \odot V^G_i)W_G^T).$$

MLP first concatenates the user embedding $U^M_u$ and the item embedding $V^M_i$ as input feature for the following multiple neural layers:

$$\hat{r}_{ui}^M = \sigma(f^{mlp}([U^M_u, V^M_i]W_M^T)),$$

where $f^{mlp}(\cdot)$ denotes the computation function for multiple neural layers.

NeuMF [7] integrates these two components by concatenating their last feature layers:

$$\hat{r}_{ui} = \sigma([U^G_u \odot V^G_i, f^{mlp}([U^M_u, V^M_i])]W^T).$$

Compared with LogisticMF, NeuMF designs a new prediction rule using the neural architectures, while its loss function remains unchanged:

$$\mathcal{L} = - \sum_{(u,i) \in \mathcal{R}^+ u \in \mathcal{R}^-} r_{ui} \ln(\hat{r}_{ui}) + (1 - r_{ui}) \ln(1 - \hat{r}_{ui}).$$

Here, $\mathcal{R}^+$ and $\mathcal{R}^-$ are the observed (user, item) pairs and the sampled unobserved (user, item) pairs, respectively. The architecture is illustrated in the left part of Figure 1.

IV. OUR SOLUTION

A. Preference Assumption

Motivated by the promising improvements achieved by the setwise preference assumption proposed in [6], we focus on refining the preference assumption on single items underlying the learning process of NeuMF. Specifically, we employ the setwise preference on both the observed and unobserved items:

$$r_{ui}P = 1, r_{uj} = 0, \; P \subseteq \mathcal{I}_u, \; j \in A, \; A \subseteq \mathcal{I}\backslash \mathcal{I}_u,$$

where $P$ is a set of observed items randomly sampled from user $u$’s history records, $A$ is a set of unobserved items randomly sampled from all the items excluding $u$’s interacted ones. Both the observed and unobserved items are organized as setwise preference. The difference is that for the observed item-sets, we feed a whole set of items into the network, since we believe that one item-set contain richer information than one single item, which may be more favorable for parameter learning of the neural network. While for the unobserved items, we input each item in a set individually. The rationale is that a set of unobserved items are subjected to more randomness.

Notice that DeepSet would not change the prediction rule of NeuMF, which means that the calculation of the predicted preference score keeps exactly the same as that of NeuMF during the prediction phase. In other words, DeepSet would not change the network architecture of NeuMF, and thus no additional model parameters will be involved.

B. DeepSet

With the preference assumption on item-sets, the key is how to integrate such a setwise preference with NeuMF to boost the leaning process. Specifically, we attempt to incorporate the effect of this new assumption by activating the setwise preference at different layers of NeuMF, namely 1) the feature input layer, 2) the feature output layer, and 3) the prediction layer, and thus we develop three variants of DeepSet on the basis of NeuMF, i.e., DeepSet(f), DeepSet(o) and DeepSet(p).

The intuition behind this layer-related activation strategy is that the earlier activations tend to capture more generalized
patterns of an item-set, while the later activations tend to keep more specialized patterns of each item in the set.

1) DeepSet(f): In this variant, the setwise preference is activated at the feature input layer, which means that we aggregate the item embeddings of the corresponding item-set, and then feed the aggregated item-set embedding into the following layers. With this variant built on GMF, we can obtain DeepSet(f, GMF), where the predicted score of each item-set would be calculated during the model learning process as follows:

\[ \hat{r}_{up}^G = \sigma((U_u^G \odot \tilde{V}_p^G)w_G^T), \]

(7)

where \( \tilde{V}_p^G \) is the aggregated embedding of the item-set \( P \) for GMF and \( \sigma \) is the sigmoid function. Like CoFiSet, we can have \( \text{DeepSet}(fi, GMF) \).

Similarly, we can get DeepSet(fi, NeuMF) by activating the average pooling on both the GMF and MLP components in NeuMF:

\[ \hat{r}_{up} = \sigma((U_u^G \odot \tilde{V}_p^G, f_{mlp}([U_u^M, \tilde{V}_p^M]))w_T), \]

(8)

where \( \tilde{V}_p^M \) is calculated as \( \tilde{V}_p^M = \frac{1}{|P|} \sum_{i \in P} V_{i}^M \) for the MLP component.

2) DeepSet(p): Different from DeepSet(f) that activates the setwise preference earlier at the feature input layer, DeepSet(p) merges the final predicted score of each item in a sampled set at the last layer of the network, i.e., the prediction layer. Specifically, each (user, item) pair in the set actually shares the whole network and would be computed in parallel, which means that we can get a set of prediction scores. The key idea is that we average these scores as the final prediction for model learning. Likewise, for GMF, we have DeepSet(p, GMF) and its predicted preference during the learning process is as follows:

\[ \hat{r}_{up}^G = \frac{1}{|P|} \sum_{i \in P} \sigma((U_u^G \odot \tilde{V}_i^G)w_G^T). \]

(9)

In a similar way, we have DeepSet(p, NeuMF) with the following predicted preference:

\[ \hat{r}_{up} = \frac{1}{|P|} \sum_{i \in P} \sigma((U_u^G \odot \tilde{V}_i^G, f_{mlp}([U_u^M, \tilde{V}_i^M]))w_T). \]

(10)

3) DeepSet(fo): Unlike DeepSet(fi) and DeepSet(p) that apply the setwise preference at the feature input layer and the prediction layer, DeepSet(fo) serves as a compromise between them, which activates the setwise average pooling operation at the last feature computation layer right before the prediction layer, i.e., the feature output layer. Therefore, we can have DeepSet(fo, GMF) and DeepSet(fo, NeuMF). Notice that DeepSet(fo, GMF) is actually equivalent to DeepSet(fi, GMF), since GMF only contains one feature computation layer. As for DeepSet(fo, NeuMF), we have:

\[ \hat{r}_{up} = \sigma((U_u^G \odot \tilde{V}_p^G, \frac{1}{|P|} \sum_{i \in P} f_{mlp}([U_u^M, \tilde{V}_i^M]))w_T). \]

(11)

These three variants actually change the learning patterns on the observed items, i.e., the individual training sample for the observed items is in the form of (user, item-set) instead of (user, item). For the unobserved items, only one (user, item) pair would be involved as an individual unobserved training sample fed to the network.

C. Loss Function

With our proposed preference assumption, the learning samples defined on the item-level have been changed to those on the set-level. Therefore, by denoting a randomly sampled observed item-set as \( P \) and the unobserved item-set as \( A \), we have the likelihood function of our DeepSet for an individual user \( u \) as:

\[ p((u, P)|\Theta)\prod_{j \in A} p((u, j)|\Theta)^{r_{uj}}, \]

(12)

where \( p((u, P)|\Theta) \) is the predicted preference of user \( u \) on item-set \( P \), which can correspond to any of the three variants proposed in Section 4.2.

Combining all users’ likelihood and taking negative log-rithm, we reach the final loss function of our DeepSet:

\[ \mathcal{L} = - \sum_{u \in U} \left( \sum_{P \subseteq I_u} r_{up} \ln(\hat{r}_{up}) + \sum_{A \subseteq I \setminus I_u} \sum_{j \in A} (1 - r_{uj}) \ln(1 - \hat{r}_{uj}) \right). \]

(13)

With a default assumption of \( r_{up} = 1 \) and \( r_{uj} = 0 \), we can rewrite the above loss function and obtain the objective function of our DeepSet:

\[ \min_{\Theta} - \sum_{u \in U} \left( \sum_{P \subseteq I_u} \ln(\hat{r}_{up}) + \sum_{A \subseteq I \setminus I_u} \sum_{j \in A} \ln(1 - \hat{r}_{uj}) \right). \]

(14)

When \( P \) and \( A \) only contain one item, the objective function reduces to the typical binary cross-entropy loss used by many deep learning-based recommendation methods. The intuition behind our DeepSet is, instead of using new structures to design elaborated prediction functions, it is more important to find a way to encourage the true power of neural networks, such as enabling more effective parameter learning by refining the preference assumptions.

D. The Analysis of Time Complexity

Since we implement our models with TensorFlow, we will analyze the time cost of DeepSet based on some operations in TensorFlow. In DeepSet, the involved computations include elementwise-product, concatenation, matrix multiplication and reduce_mean, which would be executed two times for each batch of data since there are two phases in deep learning, i.e., feedforward and backpropagation. Suppose the batch-size is \( bs \), then in the inner loop there is \( |R|^2/bs \) iterations. Denote the MLP component of NeuMF as \( n_{1} \rightarrow n_{2} \rightarrow ... \rightarrow n_{l} \), where \( n_{l} \) is the number of neurons in the \( l \)-th layer and \( l \) is the number of layers, then the computation of one iteration for the MLP part includes \( l \times n \) times of matrix multiplication. For convenience, we list the data (i.e., tensor...
TABLE I

<table>
<thead>
<tr>
<th>Methods</th>
<th>element-wise-product</th>
<th>matrix-multiplication</th>
<th>reduce_mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepSet(fi, GMF)</td>
<td>[g^b \times</td>
<td>P</td>
<td>\times d^G]</td>
</tr>
<tr>
<td>DeepSet(p, GMF)</td>
<td>[g^b \times</td>
<td>P</td>
<td>\times d^G]</td>
</tr>
<tr>
<td>DeepSet(fi, NeuMF)</td>
<td>[g^b \times</td>
<td>P</td>
<td>\times d^G] [g^b \times</td>
</tr>
<tr>
<td>DeepSet(fo, NeuMF)</td>
<td>[g^b \times</td>
<td>P</td>
<td>\times d^G] [g^b \times</td>
</tr>
<tr>
<td>DeepSet(p, NeuMF)</td>
<td>[g^b \times</td>
<td>P</td>
<td>\times d^G] [g^b \times</td>
</tr>
</tbody>
</table>

Notice that \(d^M\) is the input dimension of user and item for the GMF part, \(d^F\) is the MLP part.

TABLE II

| Dataset          | #user | #items | \(|R^T|\) \(\cup |R^{va}|\) | \(|R^{va}|\) | density     |
|------------------|-------|--------|-----------------|--------------|-------------|
| MovieLens100K    | 943   | 1,682  | 27,688          | 27,687       | 1.7456%     |
| MovieLens1M      | 6,040 | 3,953  | 287,641         | 287,640      | 1.2650%     |
| UserTag          | 3,000 | 2,000  | 123,128         | 123,218      | 2.0521%     |
| Netflix5K5K      | 5,000 | 5,000  | 77,936          | 77,936       | 0.3117%     |

We repeat this process three times to generate three copies of training data, validation data and test data for each dataset. The statistics of all the datasets are given in Table II.

B. Experimental Setting

1) Compared Methods: In order to verify whether the three variants of our DeepSet can enhance the performance of neural recommendation methods like GMF and NeuMF by refining the preference assumption with layer-related setwise preference activation strategies, we include the following state-of-the-art methods.

LogisticMF [9] is a very representative OCCF method that learns users’ preferences by minimizing a logistic loss that attempts to encourage the likelihood of an observed (user, item) pair and discourage an unobserved one.

BPR [4] is a seminal personalized ranking method that tries to maximize the preference difference between an observed item and an unobserved one for each user, which performs very well in most cases.

GMF [7] is a shallow network extended on LogisticMF, which can be viewed to replace the normal inner product of MF with a “weighted” inner product between a user and an item.

NeuMF [7] is a deep learning-based OCCF method that replaces the typical prediction rule of MF-based methods with neural architectures, while the preference assumption and the learning scheme remains the same as that in LogisticMF.

SQL-RANK [21] is a recent non-neural OCCF method that adopts a listwise preference assumption for the OCCF problem, which is widely considered as a more appropriate assumption for a ranking-oriented recommendation task.

RBM-OCCF [22, 23] uses restricted Boltzmann machines to model the one-class data, with a list of items for each user as input, i.e., the observed items and some sampled unobserved
items, which have been adopted in many pre-training tasks and shown to be very helpful for recommendation problems.

2) Evaluation Metrics: Following many empirical studies in previous studies, we use five commonly used ranking-oriented evaluation metrics, including precision, recall, F1, NDCG and 1-call. Unlike most deep learning-based recommendation methods that only rank certain number of items, we rank all the unobserved items for each user for fair comparison among all the methods. We truncate a ranking list at 5 for all the metrics, i.e., we use the performance on precision@5, recall@5, F1@5, NDCG@5 and 1-call@5 as main results.

3) Parameter Configuration: We implement LogisticMF, BPR and our DeepSet with TensorFlow\(^1\)\(\) and use Adam [24] as the optimizer. For fair comparison, we keep the dimension of the user’s latent vector and item’s latent vector used to yield inner product or element-wise product as 20. For all the methods, we tune all the hyper-parameters on the validation data. Specifically, for LogisticMF, BPR and our DeepSet, we select the learning rate from \(\{0.0001, 0.0005, 0.001, 0.005\}\) and the batch size from \(\{128, 256, 512, 1024\}\) according to the NDCG@5 performance on the validation data. For SQL-RANK and RBM-OCCF, we select the related parameters under the guidance of the original papers, which includes the learning rate, decay rate and regularization parameter. For NeuMF and our DeepSet extension upon NeuMF, the embedded multi-layer network are all fixed as 4-layer with \(64 \rightarrow 32 \rightarrow 16 \rightarrow 8\) structure, i.e., the dimension of the user’s and item’s embedding fed to MLP is 32. Following the previous work [25], the negative sampling ratio for all the methods except BPR is fixed as 3. We fix the set size at 3 for all the variants of our DeepSet.

C. Performance Comparison

Table III shows the experimental results of the classical non-neural recommendation methods (LogisticMF, BPR, SQL-RANK), neural recommendation methods with shallow and deep network (RBM, GMF and NeuMF), and the derived methods upon GMF and NeuMF with DeepSet, i.e., DeepSet(fi, GMF), DeepSet(p, GMF), DeepSet(fi, NeuMF), DeepSet(fo, NeuMF) and DeepSet(p, NeuMF), from which we can have the following observations.

- Among the four very competitive baselines, i.e., LogisticMF, BPR, SQL-RANK and RBM-OCCF, SQL-RANK performs the best on MovieLens100K and MovieLens1M, while on UserTag BPR takes a lead and RBM-OCCF shows higher accuracy on Netflix5K5K. Notice that UserTag is a relatively dense dataset and Netflix5K5K instead is very sparse, which shows that listwise preference assumption may not be friendly to very dense or sparse data, and BPR can achieve fair performance in most cases, while RBM-OCCF may be better for sparse data.
- GMF and NeuMF are actually outperformed by LogisticMF, BPR and SQL-RANK on all the four datasets, which shows that using neural networks to devise more elaborated prediction rules does not necessarily guarantee improved performance.
- All the derived methods of DeepSet upon GMF, i.e., DeepSet(fi, GMF) and DeepSet(p, GMF), significantly outperform GMF on all the datasets, which shows that our solution is very effective in boosting the performance of a shallow network like GMF. Specifically, DeepSet(fi, GMF) is more effective than DeepSet(p, GMF), which shows that activating the setwise preference at the feature input layer is more beneficial to enhance the recommendation quality for a shallow network.
- DeepSet(fi, NeuMF) performs better than NeuMF on Movielens100K and MovieLens1M, but is slightly inferior on Netflix5K5K and UserTag. Notice that DeepSet(fi, NeuMF) activates the setwise preference at the feature input layer, which may introduce more noise when the data are too sparse or too dense. This may explain the inconsistency of DeepSet(fi, NeuMF) on UserTag and Netflix5K5K, since UserTag and Netflix5K5K are the most dense and sparse datasets, respectively.
- Both DeepSet(fo, NeuMF) and DeepSet(p, NeuMF) perform substantially better than NeuMF on all the datasets, which shows that activating the setwise preference at the later layers of a deep network is more favorable to enhance the recommendation performance.
- In particular, DeepSet(fo, NeuMF) achieves the highest improvements over NeuMF, which shows that activating the setwise preference at a middle layer may be most effective in improving the performance of the deep learning-based recommendation method, especially for models like NeuMF that comprises two comparably independent components.
- Across the four datasets, the derived methods with our DeepSet indeed achieve better performance compared with those without setwise preference. Specifically, our DeepSet performs better than BPR on MovieLens100K, MovieLens1M and Netflix5K5K, and slightly weaker on UserTag. SQL-RANK is significantly better than DeepSet on MovieLens100K, but on all of the other three datasets it does not keep the good record. From this perspective, our DeepSet achieves clearly better overall performance and is highly adaptive to different datasets.

Generally speaking, DeepSet achieves comparable or even better performance than the very competitive non-neural recommendation methods, which shows that neural recommendation methods indeed have great potential to achieve remarkable performance with more appropriate preference assumptions. DeepSet(fi,GMF) performs the best among the five derived methods, which shows that improvements over a shallow network is more significant than that over a deep network with new preference assumptions on item-sets.

D. Top-k Performance Analysis

We report the performance of NDCG@k on four datasets in Figure 2, including GMF, NeuMF and our proposed solu-
TABLE III
RECOMMENDATION PERFORMANCE OF SIX BASELINE METHODS AND FIVE DERIVED METHODS OF OUR DEEPSSET UPON GMF AND NEU.MF.

<table>
<thead>
<tr>
<th>Method</th>
<th>MovieLens100K</th>
<th></th>
<th>MovieLens1M</th>
<th></th>
<th>Netflix5K5K</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision@5</td>
<td>Recall@5</td>
<td>NDCG@5</td>
<td>1-call@5</td>
<td>Precision@5</td>
<td>Recall@5</td>
</tr>
<tr>
<td>LogisticMF</td>
<td>0.3769</td>
<td>0.2964</td>
<td>0.4199</td>
<td>0.2326</td>
<td>0.2851</td>
<td>0.1959</td>
</tr>
<tr>
<td>BPR</td>
<td>0.3835</td>
<td>0.2907</td>
<td>0.4155</td>
<td>0.2451</td>
<td>0.2986</td>
<td>0.2002</td>
</tr>
<tr>
<td>SQL-RANK</td>
<td>0.3997</td>
<td>0.3111</td>
<td>0.4155</td>
<td>0.2415</td>
<td>0.3150</td>
<td>0.2322</td>
</tr>
<tr>
<td>RIM-OCCF</td>
<td>0.3792</td>
<td>0.2987</td>
<td>0.3966</td>
<td>0.2842</td>
<td>0.3050</td>
<td>0.2322</td>
</tr>
<tr>
<td>DeepSet(fi, GMF)</td>
<td>0.3853</td>
<td>0.3061</td>
<td>0.4085</td>
<td>0.2819</td>
<td>0.3111</td>
<td>0.2322</td>
</tr>
<tr>
<td>DeepSet(fo, GMF)</td>
<td>0.3825</td>
<td>0.3070</td>
<td>0.4143</td>
<td>0.2841</td>
<td>0.3150</td>
<td>0.2322</td>
</tr>
<tr>
<td>NeuMF</td>
<td>0.3501</td>
<td>0.2907</td>
<td>0.3779</td>
<td>0.2509</td>
<td>0.3050</td>
<td>0.2322</td>
</tr>
<tr>
<td>DeepSet(fi, NeuMF)</td>
<td>0.3656</td>
<td>0.2927</td>
<td>0.3799</td>
<td>0.2799</td>
<td>0.3050</td>
<td>0.2322</td>
</tr>
<tr>
<td>DeepSet(fo, NeuMF)</td>
<td>0.3659</td>
<td>0.2924</td>
<td>0.3848</td>
<td>0.2814</td>
<td>0.3086</td>
<td>0.2322</td>
</tr>
<tr>
<td>DeepSet(p, NeuMF)</td>
<td>0.3713</td>
<td>0.3041</td>
<td>0.3996</td>
<td>0.2895</td>
<td>0.3086</td>
<td>0.2322</td>
</tr>
<tr>
<td>UserTag</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Netflix5K5K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2. Top-k (NDCG) recommendation performance of DeepSet-GMF group, i.e., GMF, DeepSet(fi, GMF) and DeepSet(p, GMF), and DeepSet-NeuMF group, i.e., NeuMF, DeepSet(fi, NeuMF), DeepSet(fo, NeuMF) and DeepSet(p, NeuMF) with |P| = 3.

• For the performance of NDCG@k at different values of k, the difference between different methods keep the same as that of k = 5, which shows the relative recommendation performance are similar with different values of k. However, for all the methods, the performance of NDCG@k decreases as the value of k increases, showing that the top-k performance of recommendation models become less reliable when k gets larger.

• For the DeepSet-GMF group (i.e., GMF, DeepSet(fi,GMF) and DeepSet(p, GMF)) and the DeepSet-NeuMF group (i.e., NeuMF, DeepSet(fi, NeuMF), DeepSet(fo, NeuMF) and DeepSet(p, NeuMF)), the new models derived with DeepSet achieve the best performance of NDCG@k at different values of k (i.e., 1 ≤ k ≤ 10) on across all the four datasets, i.e., for DeepSet-GMF group, DeepSet(fi, GMF) performs the best on all the datasets, and for the DeepSet-NeuMF group, DeepSet(fo, NeuMF) performs the best on MovieLens100K, MovieLens1M and UserTag, and DeepSet(p,NeuMF) on Netflix5K5K. These results again show that our DeepSet indeed encourages the deep model (e.g., NeuMF) and the shallow model (e.g., GMF) to reach the full potential.

E. Sensitivity of the Number of Neural Layers

Figure 3 shows the performance of NeuMF and DeepSet(fo,NeuMF) with MLP of 3 layers and 4 layers on all the datasets. We can have the following observations.

• The 3-layer NeuMF performs slightly better than that with MLP of 4 layers on MovieLens100K, MovieLens1M and UserTag, while on Netflix5K5K using 4 layers are better than using 3 layers. Since the sparsity of Netflix5K5K is relatively high, we think this may show that a deeper network is more effective when the training data is highly sparse.
The performance of 3-layer DeepSet is close to DeepSet with MLP of 4 layers on all the datasets, which demonstrates that decreasing the number of neural layers does not necessarily degrade its recommendation performance. This shows that DeepSet is not very sensitive to the changes of the number of neural layers.

- DeepSet with fewer layers still significantly outperforms NeuMF with 4 layers or 3 layers, which shows the effectiveness of DeepSet upon deep learning-based recommendation models like NeuMF.

**F. Experimental Evaluation of Efficiency**

In order to evaluate the efficiency of the methods derived on GMF and NeuMF with our DeepSet, we record the training time of the five models with different set size $|\mathcal{P}|$ on the four datasets. Specifically, for all the methods, we fix the batch size as 128, learning rate as 0.0001, and number of epochs as 100, and show the results in Figure 4.

We can observe that the time cost of DeepSet(fi, GMF) and DeepSet(p, GMF) are very close at each value of $|\mathcal{P}|$, where DeepSet(fi, GMF) takes slightly more time, showing that for a shallow model like GMF, activating the setwise preference at the feature input layer cost a bit more time. For the DeepSet-NeuMF group, we can see that the CPU time of the three methods are linearly related with set size $|\mathcal{P}|$. The time of DeepSet(fo, NeuMF) and DeepSet(p, NeuMF) are very close and significantly larger than that of DeepSet(fi, NeuMF), but the time cost by all of them increase less than the $|\mathcal{P}|$ multiples, which is consistent with our analysis of the time complexity in Section IV-D. In summary, our DeepSet will not cost much more time since $|\mathcal{P}|$ is usually a small integer such as 3.

**VI. CONCLUSIONS AND FUTURE WORK**

In this paper, we propose a general solution DeepSet that refines the typical learning pattern on a single item with the preference assumption defined on item-sets. Under this new assumption, we propose to incorporate the effects of the setwise preference at the feature input layer, the feature output layer and the prediction layer of a neural model, leading to three variants, i.e., DeepSet(fi), DeepSet(fo) and DeepSet(p). Our solution effectively enhances neural recommendation methods like GMF and NeuMF. On top of the improved performance, our solution would not introduce any new model parameters.

In the future, we plan to study the effectiveness of our solution on more neural recommendation methods and more types of user behaviors [26]. Moreover, it is also interesting to explore how to activate the setwise preference in a different way.

**ACKNOWLEDGEMENT**

We thank the support of National Natural Science Foundation of China Nos. 62172283 and 61836005.

**REFERENCES**


