Learning from Sets of Items in Recommender Systems

MOHIT SHARMA, F. MAXWELL HARPER, and GEORGE KARYPIS,
University of Minnesota, USA

Most of the existing recommender systems use the ratings provided by users on individual items. An additional source of preference information is to use the ratings that users provide on sets of items. The advantages of using preferences on sets are twofold. First, a rating provided on a set conveys some preference information about each of the set's items, which allows us to acquire a user's preferences for more items than the number of ratings that the user provided. Second, due to privacy concerns, users may not be willing to reveal their preferences on individual items explicitly but may be willing to provide a single rating to a set of items, since it provides some level of information hiding. This article investigates two questions related to using set-level ratings in recommender systems. First, how users' item-level ratings relate to their set-level ratings. Second, how collaborative filtering-based models for item-level rating prediction can take advantage of such set-level ratings. We have collected set-level ratings from active users of Movielens on sets of movies that they have rated in the past. Our analysis of these ratings shows that though the majority of the users provide the average of the ratings on a set's constituent items as the rating on the set, there exists a significant number of users that tend to consistently either under- or over-rate the sets. We have developed collaborative filtering-based methods to explicitly model these user behaviors that can be used to recommend items to users. Experiments on real data and on synthetic data that resembles the under- or over-rating behavior in the real data demonstrate that these models can recover the overall characteristics of the underlying data and predict the user's ratings on individual items.

CCS Concepts: • **Information systems** → **Collaborative filtering**; **Personalization**; **Recommender systems**; • **Human-centered computing** → *User models*; *Graphical user interfaces*; *Web-based interaction*;

Additional Key Words and Phrases: Recommender systems, collaborative filtering, user behavior modeling, matrix factorization

ACM Reference format:
Mohit Sharma, F. Maxwell Harper, and George Karypis. 2019. Learning from Sets of Items in Recommender Systems. *ACM Trans. Interact. Intell. Syst.* 9, 4, Article 19 (July 2019), 26 pages.
https://doi.org/10.1145/3326128

1 INTRODUCTION

Recommender systems help consumers by providing suggestions that are expected to satisfy their tastes. They are successfully deployed in several domains such as e-commerce (e.g., Amazon, Ebay),

This work was supported in part by NSF (1447788, 1704074, 1757916, 1834251), Army Research Office (W911NF1810344), Intel Corp, and the Digital Technology Center at the University of Minnesota. Access to research and computing facilities was provided by the Digital Technology Center and the Minnesota Supercomputing Institute.
Authors' addresses: M. Sharma, F. M. Harper, and G. Karypis, University of Minnesota; emails: {sharm163, max, karypis}@umn.edu.

2160-6455/2019/07-ART19 $15.00
https://doi.org/10.1145/3326128

multimedia content providers (e.g., Netflix, Hulu), and mobile app stores (e.g., Apple, Google Play). Collaborative filtering [20, 30], which takes advantage of users' past preferences to suggest relevant items, is one of the key methods used by recommender systems.

Most collaborative filtering approaches rely on past preferences provided by users on individual items. An additional source of preferences is the user's preferences on sets of items. Examples of such set-level ratings include ratings on song playlists, music albums, reading lists, watchlists, vacation packages, product assortments, and so on. A rating provided by the user on a set of items conveys some information about the user's preference on each of the set's items and, as a result, it is a mechanism by which some information about the user's preferences can be acquired for many items. At the same time, due to privacy concerns, users that are not willing to explicitly reveal their true preferences on individual items may provide a single rating to a set of items, as it provides some level of information hiding.

This article investigates two questions related to using set-level preferences in recommender systems. First, how users' item-level ratings relate to the ratings that they provide on a set of items. Second, how collaborative filtering-based methods can take advantage of such set-level ratings towards making item-level rating predictions.

To answer the first question, we collected ratings on sets of movies from users of Movielens, a popular online movie recommender system.[1] Our analysis of these ratings leads to two key findings. First, for the majority of the users, the rating provided on a set can be accurately approximated by the average rating that they provided on the set's constituent items. Second, there is a considerable user population that tends to consistently either over- or under-rate the set, especially for sets that contain items on which the user's item-level ratings are diverse. Using these insights, we developed different models that can predict a user's rating on a set of items as well as on individual items. Furthermore, these methods can use ratings on both the sets and the items and lead to better results for the users that have either both or only one type of rating. These methods solve these problems in a coupled fashion by estimating models to predict the item-level ratings and by estimating models that combine these individual ratings to derive set-level ratings.

The key contributions of the work are the following:

(i) introduction of *Variance Offset Average Rating Model* (VOARM) and *Extremal Subset Average Rating Model* (ESARM) to model a user's consistency to over- or under-rate the set of items as a function of his/her ratings on the set's constituent items;
(ii) development of collaborative filtering-based methods that take advantage of VOARM and ESARM to estimate users' preferences on sets of items as well as on individual items; and
(iii) collection and analysis of a dataset that contains users' ratings both on individual items and on various sets containing these items.

This work significantly extends upon the preliminary work published earlier [32] by expanding the analysis of the set-based ratings by introducing a new approach to estimate the ratings from the set-level ratings (ESARM) and by expanding the experimental evaluation.

The rest of the article is organized as follows: Section 2 describes the relevant prior work. Section 3 describes the dataset creation process along with the analysis of the set ratings in relation to the users' ratings on their constituent items. Section 4 presents the methods that we developed to estimate the item-level models from the set ratings. Section 5 provides information about the evaluation methodology. Section 6 presents the results of the experimental evaluation. Section 7 provides concluding remarks.

[1] www.movielens.org.

2 RELATED WORK

There has been little published work on using set-level ratings to improve the accuracy of item-level recommendations. The one exception is a recent study in which relative preference information on different groups of items was collected during a new user signup process, and these preferences were then used to assign a user to a set of pre-built recommendation profiles [8]. This approach significantly reduced the time required to learn the user's preferences to generate recommendations for the new user. The principal difference from this approach is that, in our work, we try to model the user behavior that determines his/her estimated rating on a set and then use that to develop fully personalized recommendation methods that are not limited to new users.

Another relevant problem is of energy disaggregation [14], which refers to the task of separating the energy signal of a building into the energy signals of individual appliances that reside in the building. Disaggregated energy consumptions are used to provide feedback to consumers, forecast demands, design energy incentives, and detect appliances' malfunctions [11, 12]. Similar to the idea of energy disaggregation, in our work, we try to separate a user's rating on a set of items into the users' ratings on items in the set and generate item recommendations for the user.

Sets of items have also been used to investigate different interfaces [24] and strategies [27, 29] for preference elicitation to learn more about the users in recommender systems. Some of these techniques [27, 29] are designed to identify a set of items for which item-level ratings are then elicited by the users. Though those approaches do use sets of items, their use is not related to how they are used in the methods that we develop and study in our work. Our work requires users to provide a single rating to the set and not to its individual items.

The researchers have also investigated how different aspects, e.g., rating questions [3], reference points [1, 10, 26], and contextual factors [33], can influence a user when elicited to provide a rating on an item. In our work, we have investigated how the user provides a rating on a set of items and used the derive insights to develop collaborative filtering-based methods to predict the rating for an individual item in the set.

In addition, there has been some work that has focused on recommending lists of items or bundles of items. For example, recommendation of music playlists [2, 9, 25], travel packages [17, 21, 22, 35], reading lists [23], and recommendation of lists under user-specified budget constraints [4, 34]. However, this research is not directly related to the problems explored in this article, because our focus is on learning the user's ratings on items in lists from the ratings that the user provided on these lists.

Matrix factorization (MF) is one of the widely used collaborative filtering-based methods in recommender systems [16, 18–20]. The MF method assumes that the user-item rating matrix is low-rank and can be computed as a product of two matrices known as the user and the item latent factors. If for user u, the vector $\boldsymbol{p_u} \in \mathbb{R}^f$ denotes the f dimensional user's latent representation and similarly for item i, the vector $\boldsymbol{q_i} \in \mathbb{R}^f$ represents the f dimensional item's latent representation, then the predicted rating for user u on item i, i.e., $\hat{r}_{ui}$, is given by

$$\hat{r}_{ui} = \boldsymbol{p_u^T q_i}. \tag{1}$$

The user and item latent factors are learned by minimizing a regularized square loss between the actual and predicted ratings

$$\underset{P,Q}{\text{minimize}} \quad \frac{1}{2} \sum_{r_{ui} \in R} \left(r_{ui} - \boldsymbol{p}_u^T \boldsymbol{q}_i \right)^2 + \frac{\beta}{2} \left(||P||_F^2 + ||Q||_F^2 \right), \tag{2}$$

where the matrices $P \in \mathbb{R}^{m \times f}$ and $Q \in \mathbb{R}^{n \times f}$ contain the latent factors of the users and the items, respectively. The parameter β controls the Frobenius norm regularization of the latent factors

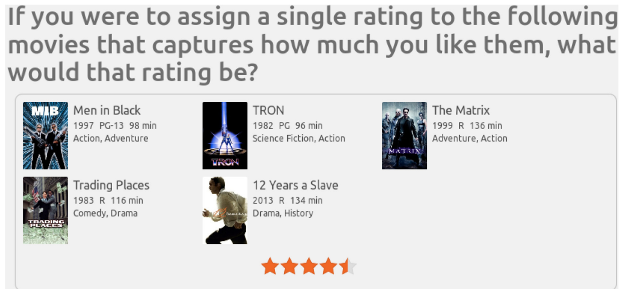

Fig. 1. The interface used to elicit users' ratings on a set of movies.

to prevent overfitting. Equation (2) can be solved by using Stochastic Gradient Descent (SGD) method [20].

3 MOVIELENS SET RATINGS DATASET

To study how the rating a user provides to a set of items relates to the ratings that the user provides on the individual items, we built a system to collect such set-level ratings and analyzed the data that were collected. The system that we developed and the analysis that we performed are described in the rest of this section. Specifically, Sections 3.1 and 3.2 describe the data collection and data pre-processing steps, respectively. Section 3.3, investigates (i) if the collected ratings are distributed uniformly or if some ratings tend to appear more than others, (ii) how a user's rating on a set relates to the user's ratings on individual movies, (iii) if the diversity of the ratings of the movies in a set could lead a user to under- or over-rate the set, (iv) whether the recently rated items carry more weight than the items rated a long time ago, (v) if the difference in the content of the items in a set could lead a user to under- or over-rate the set; and (vi) if there are users that tend to consistently under- or over-rate sets.

3.1 Data Collection

Movielens is a recommender system that utilizes collaborative filtering algorithms to recommend movies to their users based on their preferences. Figure 1 shows the set rating widget that we developed to obtain ratings on a set of movies from the Movielens users. The set rating widget could be rated from 0.5 to 5 with a precision of 0.5. For the purpose of data collection, we selected users who were active since January 2015 and have rated at least 25 movies. The selected users were encouraged to participate by contacting them via email. The sets of movies that we asked a user to rate were created by selecting five movies at random without replacement from the movies that they have already rated. Hence, the user was familiar with the movies in the set that we asked him/her to rate. Furthermore, we limited the number of sets a user can rate in a session to 50, though users can potentially rate more sets in different sessions. The set rating widget went live on February 2016 and, for the purpose of this study, we used the set ratings that were provided until April 2016.

3.2 Data processing

From the initially collected data, we removed users who have rated sets within a time interval of less than 1s to avoid users who might be providing the ratings at random. After this pre-processing, we were left with ratings from 854 users over 29,516 sets containing 12,549 movies. This dataset, after pre-processing, is available publicly to the community for further research.[2] Figure 2 shows

[2] https://grouplens.org/datasets/learning-from-sets-of-items-2019/.

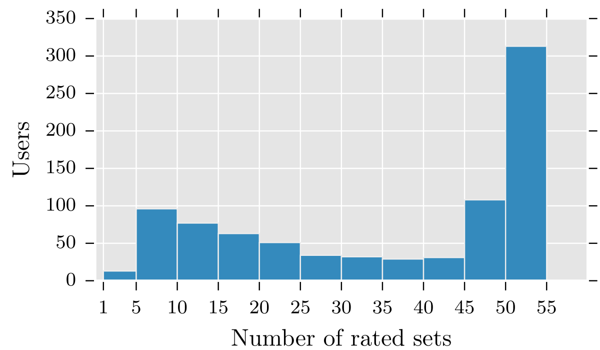

Fig. 2. The distribution of number of sets rated by the users.

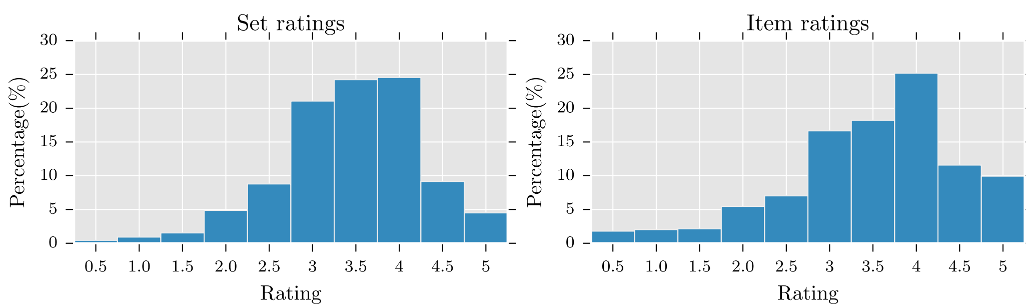

Fig. 3. The distribution of the provided set ratings (left) and the ratings of their constituent items (right).

the distribution of the number of sets rated by the users, which shows that roughly half of the users have rated at least 45 sets in a session.

3.3 Analysis of the Set Ratings

We investigated whether ratings are distributed uniformly or if some ratings tend to appear more than others. Figure 3 (left) depicts the distribution of the collected ratings over all the sets. The majority of the ratings lie between 3.0 and 4.0, since, by construction, we know the actual ratings that these users provided on the actual movies. Figure 3 (right) shows the distribution of the ratings of the movies that were contained in all these sets. By comparing these distributions, we can see that the average item-rating (3.50) is somewhat higher than the average set-based rating (3.44), but the overall variance of the set-based ratings (0.65) is lower than that of the item ratings (1.01).

To analyze how consistent a user's rating on a set is with the ratings provided by the user on the movies in the set, we computed the difference of the average of the user's ratings on the items in the set and the rating assigned by a user to the set. We will refer to this difference as *mean rating difference* (MRD). Figure 4 (left) shows the distribution of the MRD values in our datasets. The majority of the sets have an MRD within a margin of 0.5, indicating that the users have rated them close to the average of their ratings on the set's items. The remaining of the sets have been rated either significantly lower or higher than the average rating. We refer to these sets as the under- and the over-rated sets, respectively. Moreover, an interesting observation from the results in Figure 4 (left), is that the number of under-rated sets is more than that of the over-rated sets.

To understand what can lead to a set being under- or over-rated, we investigated if the *diversity* of the ratings of the individual movies in a set could lead a user to under- or over-rate the set. We

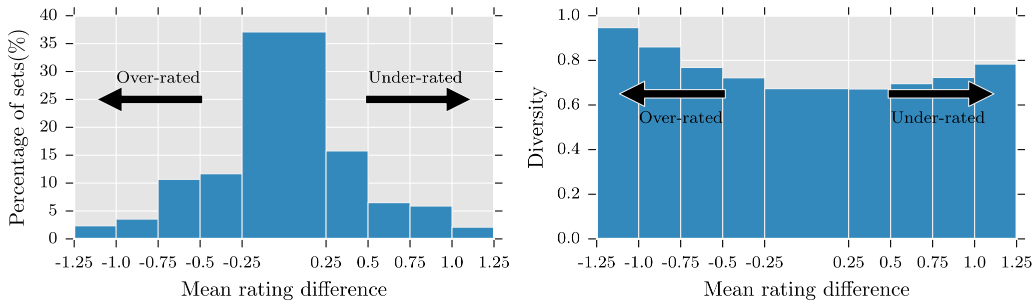

Fig. 4. Histogram of percentage of sets (left) and diversity (right) against mean rating difference (MRD).

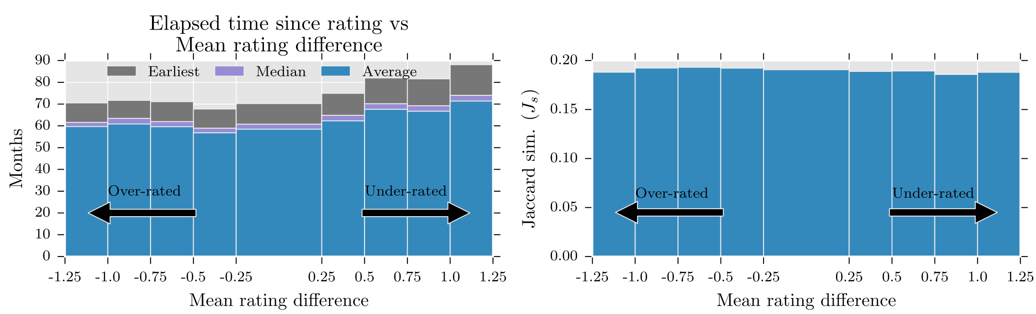

Fig. 5. Histogram of elapsed time in months (left) and jaccard similarity of movies (right) against mean rating difference (MRD).

measured the diversity of a set as the standard deviation of the ratings that a user has provided to the individual items of the set. As shown in Figure 4 (right), the sets that contain more diverse ratings (i.e., higher standard deviations) tend to get under- or over-rated more often when compared to less diverse sets. This trend was found to be statistically significant (p-value of 0.01 using t-test).

Furthermore, we investigated whether the recently rated items carry more weight than the items rated a long time ago. To this end, we computed the difference between the timestamp of the earliest rating of the movies in the set and the year 2016, i.e., when the users were asked to rate the sets. Similarly, we computed the median and average age of movies in a set. Interestingly as shown in Figure 5 (left), the under-rated sets contained movies whose ratings were provided on average five years before the survey, while the remaining sets contained the movies whose ratings were provided on average four years before the survey. This difference among the sets was found to be statistically significant (p-value < 0.001 using t-test). This suggests that the user's preference for a movie rated in the past carries lower weight than the recently rated movie. The user's higher preference for a recent movie is not surprising, as it has been shown that the user tends to rate a movie close to the middle of the scale as the time between viewing a movie and rating it increases [5].

Moreover, the difference in the content of the items in a set may also lead a user to under- or over-rate the set. We examined if the difference in genres of movies in a set can lead to under- and over-rating of the set. To this end, we computed average pairwise *jaccard* similarity of the movies in a set after considering genres of the movies in the set. Average pairwise jaccard similarity (J_s)

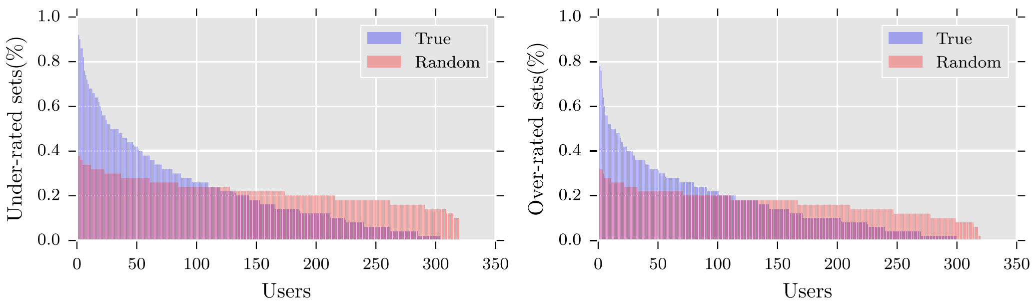

Fig. 6. Fraction of under-rated and over-rated sets across users in true and random population.

of the movies in set $\mathcal{S}$ is given by

$$J_s = \frac{2}{|\mathcal{S}|(|\mathcal{S}|-1)} \sum_{i=1}^{|\mathcal{S}|} \sum_{j=i+1}^{|\mathcal{S}|} \frac{|\mathcal{G}_i \cap \mathcal{G}_j|}{|\mathcal{G}_i \cup \mathcal{G}_j|}, \tag{3}$$

where $|\mathcal{S}|$ denotes size of set $\mathcal{S}$ and $\mathcal{G}_k$ represents the set of genres of movie k in set $\mathcal{S}$. Interestingly, as can be seen in Figure 5 (right), the average jaccard similarity of movies in sets is comparable across the under- or over-rated sets, and the variation of jaccard similarity was found not to be statistically significant (p-value of 0.769 using t-test). The insignificant variation in jaccard similarity suggests that a user rating on a set of movies is not influenced by the difference in genres of the movies in the datset.

Additionally, we studied if there are users that tend to consistently under- or over-rate sets. To this end, we selected users who have rated at least 50 sets and computed the fraction of their under- and over-rated sets. We also computed the fraction of under- and over-rated sets across a random population of the same size. We generated this random population by randomly permuting the under-rated and over-rated sets across the users. Figure 6 shows the fraction of under- and over-rated sets for both the true and random population of user. In the true population, some users tend to under- or over-rate sets significantly more than that of the random population. Using the Kolmogorov-Smirnov 2 sample test, we found this behavior of true population to be statistically different (p-value < 0.001) from that of random population.

The above analysis reveals that our dataset contains users that when they are asked to assign a single rating to a set of items, some of them consistently assign a rating that is lower than the average of the ratings that they provided to the set's constituent items (they under-rate), whereas others assign a rating that is higher (they over-rate). Thus, some users are very demanding (or picky) and tend to focus on the worst items in the set; whereas other users are less demanding and tend to focus on the best items in the set. Henceforth, we will refer to the tendency of a user to under-rate sets of items as the user-specific *pickiness*. Moreover, we will refer to a user as being picky if the user under-rates a set and less picky if the user over-rates the set.

4 METHODS

In this section, we investigate different approaches that capture the user behavior of providing ratings on sets. We describe various methods that use the set ratings alone or in combination with individual item ratings towards solving two problems: (i) predict a rating for a set of items, and (ii) predict a rating for individual items. Our methods solve these problems in a coupled fashion by estimating models for predicting the ratings that users will provide to the individual items and by estimating models that use these item-level ratings to derive set-level ratings.

4.1 Modeling Users' Ratings on Sets

To estimate the preferences on individual items from the preferences on the sets, we need to make some assumptions on how a user derives a set-level rating from the ratings of the set's constituent items. Informed by our analysis of the data described in Section 3, we investigated three approaches of modeling.

Average Rating Model (ARM). The first approach assumes that the rating that a user provides on a set reflects his/her average rating on all the items in the set. Specifically, if the rating of user u on set $\mathcal{S}$ is denoted by r_u^s, then the estimated rating of user u on set $\mathcal{S}$ is given by

$$\hat{r}_u^s = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} r_{ui}. \tag{4}$$

As the analysis in Section 3 showed, such a model correlates well with the actual ratings that the users provided on the majority of the sets, especially when the ratings of the constituent items are not very different.

Extremal Subset Average Rating Model (ESARM). To capture the user-specific *pickiness*, i.e., the tendency of a user to under-rate sets of items, illustrated in Figures 4 and 6, this approach postulates that a user rates a set by considering only a subset of the set's items. If a user tends to consistently under-rate each set, then that subset will contain some of each set's lowest-rated items. Analogously, if a user tends to consistently over-rate each set, then that subset will contain some of each set's highest-rated items. Moreover, this approach further postulates that given such subsets, the rating that a user will assign to the set as a whole will be the average of his/her ratings on the individual items of the subset. The parameter in this model that captures the level of a user's pickiness is the size of the subset and whether or not it will contain the least- or the highest-rated items. We will call these subsets having least- and highest- rated items as *extremal subsets*. The set rating of an extremely picky user will be determined by the average rating of one or two of the least-rated items, whereas the set rating of a user that is not picky at all will be determined by the average rating of one or two of the highest-rated items.

If e_i denotes the average rating of items in ith extremal subset and n_s denotes the number of items in set $\mathcal{S}$, then $\langle e_1, \ldots, e_{n_s}, \ldots, e_{2n_s-1} \rangle$ represents the average rating on all the extremal subsets; for $1 \leq i \leq n_s$, e_i is the average rating of i least-rated items, for $n_s \leq i \leq 2n_s - 1$, e_i is the average rating of the $2n_s - i$ highest-rated items, and e_{n_s} is the average rating of all the items in the set. Then, $\hat{r}_u^s$ is given by

$$\hat{r}_u^s = \sum_{i=1}^{2n_s-1} w_{u,i} e_i, \tag{5}$$

where $w_{u,i}$ is a non-negative weight of user u on ith extremal subset and the weights sum to 1. The weight $w_{u,i}$ measures the influence of the items in ith extremal subset towards estimating the user's rating on set $\mathcal{S}$. One of the weights corresponds to the extremal subset that is responsible for majority of the user's rating on set, and it is higher than others, i.e.,

$$\begin{aligned}
&\sum_{i=1}^{2n_s-1} w_{u,i} = 1, \\
&w_{u,j} < w_{u,j+1}, \forall j < k, \\
&w_{u,j+1} < w_{u,j}, \forall j \geq k, \\
&w_{u,k} > c, c > 0,
\end{aligned} \tag{6}$$

where c is the minimum weight of the extremal subset having the highest contribution towards the user's rating on the set.

Note that this model assumes that the size of all the sets are the same; however, it can be generalized to sets of different sizes by constructing the extremal subsets for a fixed number of quantiles in a set.

Variance Offset Average Rating Model (VOARM). This approach captures the user-specific *pickiness* by assuming that a user rates a set by considering both the average rating of the items in the set and also the diversity of the set's items. In this model, the set's rating is determined as the sum of the average rating of the set's items and a quantity that depends on the set's diversity (e.g., the standard deviation of the set's ratings) and the user's level of pickiness. If a user is very picky, that quantity will be negative and large, resulting in the set being (severely) under-rated.However, if a user is not picky at all, that quantity will be positive and large, resulting in the set being (severely) over-rated.

If β_u denotes the pickiness level of user u, then the estimated rating on a set is given by

$$\hat{r}_u^s = \mu_s + \beta_u \sigma_s, \tag{7}$$

where μ_s and σ_s are the mean and the standard deviation of the ratings of items in the set $\mathcal{S}$. Both μ_s and σ_s are given by

$$\mu_s = \frac{1}{|S|} \sum_{i \in \mathcal{S}} r_{ui}, \quad \sigma_s = \sqrt{\frac{1}{|S|} \sum_{i \in \mathcal{S}} (r_{ui} - \mu_s)^2}. \tag{8}$$

In contrast to ESARM, this method considers all the items in a set by considering the average rating of items in the set, i.e., μ_s, and the user's level of pickiness, i.e., β_u, determines how a user's rating on the set is affected by the least-rated movies or the highest-rated movies in the set.

4.2 Modeling User's Ratings on Items

To model a users' ratings on the items, similar to matrix factorization method [20], we assume that the underlying user-item rating matrix is low-rank, i.e., there is a low-dimensional latent space in which both the users and the items can be compared to each other. The rating of user u on item i can be computed as an inner product of the user and the item latent factors in that latent space. Thus, the estimated rating of user u on item i, i.e., $\hat{r}_{ui}$, is given by

$$\hat{r}_{ui} = \boldsymbol{p}_{\boldsymbol{u}}^T \boldsymbol{q}_{\boldsymbol{i}}, \tag{9}$$

where $\boldsymbol{p}_{\boldsymbol{u}} \in \mathbb{R}^f$ is the latent representation of user u, $\boldsymbol{q}_{\boldsymbol{i}} \in \mathbb{R}^f$ is the latent representation of item i, and f is the dimensionality of the underlying latent space.

4.3 Combining Set and Item Models

Our goal is to estimate the item-level ratings by learning the user and item latent factors of Equation (9); however, the ratings that we have available from the users are at the set-level. To use the available set-level ratings, we need to combine Equation (9) with Equations (4), (5), and (7). To solve the problem, we assume that the actual item-level ratings used in Equations (4), (5), and (7) correspond to the estimated ratings given by Equation (9). Hence, the estimated set-level ratings in Equations (4), (5), and (7) are finally expressed in terms of the corresponding user and item latent factors.

ALGORITHM 1: Learn ARM

procedure LearnARM
  $\eta \leftarrow$ learning rate
  $\lambda \leftarrow$ regularization parameter
  $\mathcal{R}^s \leftarrow$ all users' ratings on sets
  $iter \leftarrow 0$
  Init P, Q with random values $\in [0,1]$
  while iter < maxIter and RMSE on validation set decreases **do**
    $\mathcal{R}^s \leftarrow \text{shuffle}(\mathcal{R}^s)$
    for all $r_u^s \in \mathcal{R}^s$ **do**
      $\hat{r}_u^s \leftarrow \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \boldsymbol{p}_u^T \boldsymbol{q}_i$
      $e_u^s \leftarrow (\hat{r}_u^s - r_u^s)$
      $\boldsymbol{v}_k \in \mathbb{R}^k \leftarrow 0$
      for each item $i \in s$ **do**
        $\boldsymbol{v}_k \leftarrow \boldsymbol{v}_k + \boldsymbol{q}_i$
      end for
      $\boldsymbol{p}_u \leftarrow \boldsymbol{p}_u - \eta(\frac{e_u^s}{|s|}\boldsymbol{v}_k + \lambda \boldsymbol{p}_u)$ ▷ Update user's latent representation
      for each item $i \in s$ **do**
        $\boldsymbol{q}_i \leftarrow \boldsymbol{q}_i - \eta(\frac{e_u^s}{|s|}\boldsymbol{p}_u + \lambda \boldsymbol{q}_i)$ ▷ Update item's latent representation
      end for
    end for
    $iter \leftarrow iter + 1$
  end while
end procedure

4.4 Model Learning

The parameters of the models that estimate item- and set-level ratings are the user and item latent vectors (p_u and q_i): in the case of ESARM method, the users' weights on extremal subsets (W); and in the case of the VOARM method, the user's pickiness level (β_u). These parameters are estimated using the user-supplied set-level ratings by minimizing a square error loss function given by

$$\mathcal{L}_{rmse}(\Theta) \equiv \sum_{u \in U} \sum_{s \in \mathcal{R}_u^s} (\hat{r}_u^s(\Theta) - r_u^s)^2, \tag{10}$$

where Θ represents model parameters, U represents all the users, $\mathcal{R}_u^s$ contains all the sets rated by user u, r_u^s is the original rating of user u on set $\mathcal{S}$, and $\hat{r}_u^s$ is the estimated rating of user u on set $\mathcal{S}$.

To control model complexity, we add regularization of the model parameters, thereby leading to an optimization process of the following form:

$$\underset{\Theta}{\text{minimize}}\ \mathcal{L}_{rmse}(\Theta) + \lambda(||\Theta||^2), \tag{11}$$

where λ is the regularization parameter. The L2-regularization is added to reduce the model complexity, thereby improving its generalizability. This optimization problem can be solved by the Stochastic Gradient Descent (SGD) [6] algorithm.

Note that for the ESARM model, we need to solve this optimization problem with linear and non-negative constraints on user weights $\boldsymbol{w}_u^T$. If we know the users' and the items' latent factors, then the user weights can be determined by solving Equation (11) as a constraint quadratic programming [7] for each user. We can determine a user's weights by solving multiple quadratic programs, each corresponding to a different extremal subset having the highest weight, and selecting the solution that has lowest RMSE over the user's sets. Hence, for ESARM, we solve for W and $\{\boldsymbol{p}_u, \boldsymbol{q}_i\}$ alternately at each SGD iteration. In ESARM, the minimum weight of the extremal

ALGORITHM 2: Learn ESARM

procedure LEARN*ESARM*
  $\eta \leftarrow$ learning rate
  $\lambda \leftarrow$ regularization parameter
  $\mathcal{R}^s \leftarrow$ all users' ratings on sets
  $n_s \leftarrow$ number of items in set
  $n_{es} \leftarrow 2n_s - 1$ ▷ number of possible extremal subsets
  $iter \leftarrow 0$
  Init P, Q with random values $\in [0,1]$
  Init W with random values $\forall$ user $u \in U$, s.t., $\sum_{i=1}^{n_{es}} w_{u,i} = 1$
  while iter < maxIter and RMSE on validation set decreases **do**
    $\mathcal{R}^s \leftarrow \text{shuffle}(\mathcal{R}^s)$

    for all $r_u^s \in \mathcal{R}^s$ **do**
      $\hat{r}_u^s \leftarrow 0$
      $\mathcal{E}_s \leftarrow$ All possible extremal subsets for set $\mathcal{S}$

      $\nabla p_u \in \mathbb{R}^f \leftarrow 0$
      for each subset $i \in \mathcal{E}_s$ **do**
        $\hat{e}_i \leftarrow 0,\ q_{sum} \in \mathbb{R}^f \leftarrow 0$
        for each item $j \in i$ **do**
          $\hat{e}_i \leftarrow \hat{e}_i + p_u^T q_j,\ q_{sum} \leftarrow q_{sum} + q_j$
        end for
        $\hat{e}_i \leftarrow \frac{\hat{e}_i}{|i|},\ q_{sum} \leftarrow \frac{q_{sum}}{|i|}$
        $\hat{r}_u^s \leftarrow \hat{r}_u^s + w_{u,i}\hat{e}_i$
        $\nabla p_u \leftarrow \nabla p_u + w_{u,i} q_{sum}$
      end for
      $e_u^s \leftarrow (\hat{r}_u^s - r_u^s)$
      $\nabla p_u \leftarrow 2e_u^s \nabla p_u + 2\lambda p_u$ ▷ update user's latent representation
      $p_u \leftarrow p_u - \eta \nabla p_u$

      $\nabla q \leftarrow 2e_u^s p_u$
      for each subset $i \in \mathcal{E}_s$ **do**
        for each item $j \in i$ **do**
          $q_j \leftarrow q_j - \eta(\frac{w_{u,i}\nabla q}{|i|} + 2\frac{\lambda}{n_s} q_j)$ ▷ update items' latent representation
        end for
      end for
    end for

    for all $u \in U$ **do**
      Update $\boldsymbol{w}_u$ using constraint quadratic programming as described
      in Section 4.4.
    end for

    $iter \leftarrow iter + 1$
  end while
end procedure

ALGORITHM 3: Learn *VOARM*

procedure LearnVOARM
    $\eta \leftarrow$ learning rate
    $\lambda \leftarrow$ regularization parameter
    $\mathcal{R}^s \leftarrow$ all users' ratings on sets
    $iter \leftarrow 0$
    Init P, Q and βs with random values $\in [0,1]$
    while iter < maxIter and RMSE on validation set decreases **do**
        $\mathcal{R}^s \leftarrow \text{shuffle}(\mathcal{R}^s)$
        for all $r_u^s \in \mathcal{R}^s$ **do**
            $\hat{\mu}_s \leftarrow \frac{1}{|\mathcal{S}|}\sum_{i \in \mathcal{S}} \boldsymbol{p}_u^T \boldsymbol{q}_i$
            $\hat{\sigma}_s \leftarrow \epsilon + \sqrt{\frac{1}{|\mathcal{S}|}\sum_{i \in \mathcal{S}} (\boldsymbol{p}_u^T \boldsymbol{q}_i - \hat{\mu}_s)^2}$
            $\hat{r}_u^s \leftarrow \hat{\mu}_s + \beta_u \hat{\sigma}_s$
            $e_u^s \leftarrow (\hat{r}_u^s - r_u^s)$
            $\boldsymbol{q} \in \mathbb{R}^f \leftarrow 0,\ \boldsymbol{v} \in \mathbb{R}^f \leftarrow 0$
            for each item $i \in \mathcal{S}$ **do**
                $\boldsymbol{q} \leftarrow \boldsymbol{q} + \boldsymbol{q}_i$
                $\boldsymbol{v} \leftarrow \boldsymbol{v} + (\boldsymbol{p}_u^T \boldsymbol{q}_i)\boldsymbol{q}_i$
            end for
            $\nabla p_u \leftarrow \frac{\boldsymbol{q}}{|\mathcal{S}|} + \frac{\beta_u \boldsymbol{v}}{\hat{\sigma}_s |\mathcal{S}|} - \frac{\beta_u \mu_s \boldsymbol{q}}{\hat{\sigma}_s |\mathcal{S}|}$
            $\nabla q \leftarrow \frac{2 e_u^s p_u}{\mathcal{S}}$
            for each item $i \in \mathcal{S}$ **do**
                $t \leftarrow 1 + \frac{\beta_u p_u^T q_i}{\hat{\sigma}_s} - \frac{\beta_u \mu_s}{\hat{\sigma}_s}$
                $q_i \leftarrow q_i - \eta(t \nabla q + 2\lambda q_i)$ ▷ Update item's latent representation
            end for
            $p_u \leftarrow p_u - \eta(2 e_u^s \nabla p_u + 2\lambda p_u)$ ▷ Update user's latent representation
            $\beta_u \leftarrow \beta_u - \eta(2 e_u^s \hat{\sigma}_s + 2\lambda \beta_u)$ ▷ Update β_u
        end for
        $iter \leftarrow iter + 1$
    end while
end procedure

subset having highest contribution towards ratings on sets, i.e., c, can be specified in the range [0,1]. Also, in the VOARM method, we add a fixed constant, i.e., ϵ in [0, 1], to computed σ for robustness. Algorithms 1, 2, and 3 show the steps used to learn the ARM, ESARM, and VOARM models, respectively.

If we also have ratings for the individual items, then we can incorporate these ratings into the model estimation process by treating each item as a set of size one. Note that, when we do not have set-level ratings but only have item-level ratings, then the proposed methods reduce to MF, as we need to estimate a single item-level rating to estimate the set-level rating.

5 EXPERIMENTAL EVALUATION

5.1 Dataset

We evaluated the proposed methods on two datasets: (i) the dataset analyzed in Section 3, which will be referred to as ML-RealSets, and (ii) a set of synthetically generated datasets that allow us to assess how well the optimization algorithms can estimate accurate models and how their accuracy depends on various data characteristics.

The synthetic datasets were derived from the Movielens 20M dataset[3] [13], which contains 20M ratings from approximately 229,060 users on 26,779 movies. For experiment purposes, we created a synthetic low-rank matrix of rank 5 as follows: We started by generating two matrices $A \in \mathbb{R}^{n \times k}$ and $B \in \mathbb{R}^{m \times k}$, where n is number of users, m is number of items, and $k = 5$, whose values are uniformly distributed at random in $[0, 1]$. We then computed the singular value decomposition of these matrices to obtain $A = U_A \Sigma_A V_A^T$ and $B = U_B \Sigma_B V_B^T$. We then let $P = \alpha U_A$, $Q = \alpha U_B$ and $R = PQ^T$. Thus, the final rank k matrix R is obtained as the product of two randomly generated rank k matrices whose columns are orthogonal. Note that the parameter α was determined empirically to produce ratings in the range of $[-10, 10]$.

Since we know the complete synthetic low-rank matrix, we can generate the rating corresponding to an observed user-item pair in the real dataset from the complete rating matrix. We randomly selected 1,000 users without replacement from the dataset, and for each user we created sets containing five movies. The movies in a user's set were selected at random without replacement from the movies rated by that user. For each user, we created at least k such sets of movies, where $k \in [40, 60, 80, 100, 140]$. We generated ratings for a user on a set by following two approaches:

(i) ESARM-based rating: For each user, we chose one of the extremal subsets at random and used that to generate ratings for all the sets. The set is assigned an average of the user's ratings on the items in the chosen extremal subset of the items in the set.
(ii) VOARM-based rating: For each user, we chose the user's level of pickiness (the β_u parameter) at random from the range $[-2.0, 2.0]$. The set is assigned an average of the user's ratings on the items in the set, and also we offset this rating by adding a quantity computed by scaling the standard deviation of ratings in the set by the randomly chosen user's level of pickiness.

For all these datasets, we added random $\mathcal{N}(0, 0.1)$ Gaussian noise while computing ratings at both the item and set-level for the users. For each approach, we generated 15 different synthetic datasets, each by varying the user-item latent factors and the users' pickiness.

5.2 Evaluation Methodology

To evaluate the performance of the proposed methods, we divided the available set-level ratings for each user into training, validation, and test splits by randomly selecting five set-level ratings for each of the validation and test splits. The validation split was used for model selection. To assess the performance of the methods for item recommendations, we used a test set that contained for each user the items that were not present in the user's sets (i.e., these were absent from the training, test, and validation splits) but were present in the original user-item rating matrix used to generate the sets. We used Root Mean Square Error (RMSE) to measure the accuracy of the rating prediction over items and sets.

5.2.1 Comparison Methods. In addition to the evaluation of the proposed methods, i.e., ARM, ESARM, and VOARM, we also present the results for the following methods:

(i) SetAvg: This personalized method predicts a user's ratings on items and sets as the average of the user's ratings on sets. The rating of user u on set $\mathcal{S}$ is given by

$$\hat{r}_u^s = \frac{1}{|Q_u|} \sum_{k \in Q_u} r_{uk}, \tag{12}$$

[3]https://grouplens.org/datasets/movielens/20m/.

where $\mathcal{Q}_u$ represents all the sets rated by user u. The rating of user u on item i is given by

$$\hat{r}_{ui} = \frac{1}{|\mathcal{Q}_u|} \sum_{k \in \mathcal{Q}_u} r_{uk}, \tag{13}$$

where $\mathcal{Q}_u$ represents all the sets rated by user u.

(ii) Item average: This non-personalized method estimates the rating for an item as the average of the ratings provided by the users on the item. The rating $\hat{r}_i$ for an item i is given by

$$\hat{r}_i = \frac{1}{|\mathcal{U}_i|} \sum_{u \in \mathcal{U}_i} r_{ui}, \tag{14}$$

where $\mathcal{U}_i$ denotes the set of users who have rated item i.

(iii) UserMeanSub: This non-personalized method estimates the rating for an item as the sum of average rating on sets and average of user mean subtracted item ratings. The rating $\hat{r}_i$ for an item i is given by

$$\hat{r}_i = \mu_s + \frac{1}{|\mathcal{U}_i|} \sum_{u \in \mathcal{U}_i} \left(r_{ui} - \frac{1}{|\mathcal{I}_u|} \sum_{k \in \mathcal{I}_u} r_{uk} \right), \tag{15}$$

where μ_s is the average of the ratings on all the sets; $\mathcal{I}_u$ represents the set of items rated by user u.

(iv) MFSET: This personalized method assumes that a user's ratings on the items in a set are equal to the rating provided by the user on the set. It uses these item-level ratings to estimate the user and the item latent factors by using the MF method. The rating of user u on item i is given by

$$\hat{r}_{ui} = \boldsymbol{p_u}^T \boldsymbol{q_i}, \tag{16}$$

where the vector $\boldsymbol{p_u} \in \mathbb{R}^f$ denotes the f dimensional user's latent representation and similarly for item i, the vector $\boldsymbol{q_i} \in \mathbb{R}^f$ represents the f dimensional item's latent representation.

(v) MFOpt: This method uses the actual users' ratings on the items in the set to estimate the user and the item latent factors by using the MF method. Note that given that MFOpt uses the actual item-level ratings, its performance will be better than the other methods that rely only on set-level ratings. As such, MFOpt's performance can be used to assess the *opportunity cost* associated with using set-level ratings over using the corresponding item-level ratings.

In practice, a significant proportion of the ratings provided by users on items depends on factors that are associated with either users or items and do not depend on interactions between the users and the items. For example, some users have a tendency to rate higher than others, and some items tend to receive higher ratings than others. For the real set-level rating dataset we obtained from Movielens users, we model these tendencies by estimating user- and item-biases [20] as part of the model learning.

5.3 Model Selection

We performed grid search to tune the dimensions of the latent factors and regularization hyper-parameters for the latent factors. We searched for regularization weights (λ) in the range [0.001, 0.01, 0.1, 1, 10], ϵ in the range [0.1, 0.25, 0.5, 1], and c in the range [0, 0.25, 0.50, 0.75, 0.90] for both the synthetic and the real datasets. We searched for the dimension of latent factors (f) in the range [1, 5, 10, 15, 25, 50, 75, 100] for real datasets, and used 5 as the dimension of latent factors for synthetic datasets. The final parameters were selected based on the performance on the validation split.

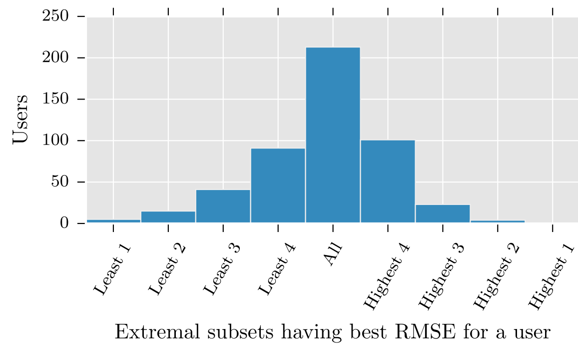

Fig. 7. The number of users for which their pickiness behavior is explained by the corresponding least- and highest-rated subsets of items.

6 RESULTS AND DISCUSSION

The experimental evaluation of the various methods that we developed is done in three phases. First, we investigated how well the proposed models can explain the users' ratings over sets in the dataset we obtained from a subset of Movielens users (described in Section 3). Second, we evaluated the performance of the methods using the synthetically generated datasets to assess how well the underlying optimization algorithms can recover the underlying data generation models and achieve good prediction performance at either the set- or item-level. Note that unless otherwise specified, we report the average of RMSEs of all the synthetic datasets as the final RMSE values for each rank and proposed approach. Finally, we evaluated the prediction performance achieved by the proposed methods at both the set- or item-level in the real dataset.

6.1 Agreement of Set-rating Models with the Observed Data

To determine how well the proposed models can explain the ratings that the users in our dataset provided, we performed the following analysis: We selected sets with standard deviation of at least 0.5 and included only those users who have rated at least 20 such sets. This left us with 17,552 sets rated by 493 users.

To study the ESARM model, for each set rated by a user, we created all the possible subsets having either k lowest- or k highest-rated items for all the possible values of $k \in [1, 5]$, i.e., nine extremal subsets. We computed the error between the average rating of items in the extremal subsets and the rating provided by a user on a set. Similarly, we computed the error over the remaining sets for a user and selected that subset among the nine extremal subsets corresponding to which the user has lowest Root Mean Square Error (RMSE) for all the sets. Figure 7 shows the number of users and their corresponding extremal subset that obtained lowest RMSE for their sets. As can be seen in the figure, there are certain users for whom the lowest RMSE on sets corresponds to either k lowest- or k highest-rated items in a set, where $k < 5$. This indicates that while providing a rating to a set of items, the user may get influenced more by a subset of the items in a set rather than all the items in the set.

Further, to investigate the VOARM model, we computed the user's level of pickiness (β_u) as

$$\beta_u = \frac{1}{n_s} \sum_{s=1}^{n_s} \frac{r_u^s - \mu_s}{\sigma_s}, \tag{17}$$

where n_s is the number of sets rated by user u, r_u^s denotes the rating provided by user u on set s, μ_s is the mean rating of the items in set s, and σ_s is the standard deviation of the ratings of the

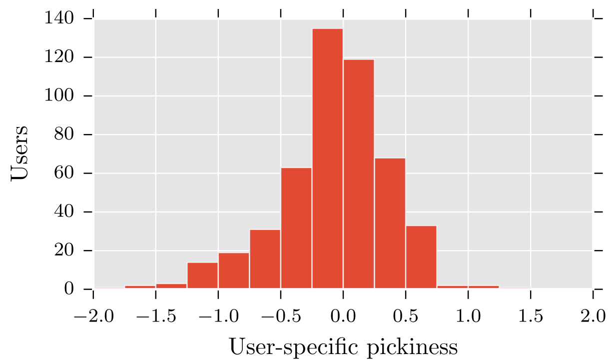

Fig. 8. The number of users and their computed level of pickiness.

Table 1. Fit of Different Rating Models on the Data

	ARM	ESARM	VOARM
RMSE	0.597	0.509	0.521

items in set s. Figure 8 shows the histogram of the users' level of pickiness. As can be seen from the figure, certain users tend to under- or over-rate sets with high standard deviation. We conducted a t-test on the magnitude of values of pickiness between the set of users that under-rate sets and the set of users that over-rate sets. We found the behavior of users under- and over-rating sets to be statistically significant (p-value < 0.001 using t-test). Furthermore, we observe that more users (268) tend to under-rate sets than over-rate them (224).

Additionally, we computed how well the above rating models, i.e., ESARM and VOARM, compare against the ARM model, where a user rates a set as the average of the ratings that he/she gives to the set's items. We used the user-specific pickiness determined in the above analysis for the ESARM and the VOARM models to estimate a user's rating on a set. Table 1 shows the RMSE of the estimated ratings according to different models, and as can be seen in the table, both the ESARM and the VOARM give a better fit to the real data than ARM, thereby suggesting that modeling users' level of pickiness could lead to better estimates.

6.2 Performance on the Synthetic Datasets

6.2.1 Accuracy of Set- and Item-level Predictions. We investigated the performance of the proposed methods for both item- and set-level predictions on the synthetic datasets. In addition to the performance of each method on its corresponding dataset, we also show the performance of the ARM and SetAvg methods in Figures 9 and 10.

Figure 9 shows that ESARM outperforms all other methods for both set- and item-level predictions for datasets with a large number of sets. However, for datasets with fewer sets, ARM outperforms ESARM and SetAvg for the set- and item-level predictions. Additionally, ARM outperforms MFSET for item-level predictions as well. Figure 10 shows that VOARM outperforms all other methods for both set- and item-level predictions. Unlike ESARM, VOARM performs better than other methods even for the case when we have fewer sets, and this suggests that ESARM needs a larger number of sets than VOARM to recover the underlying characteristics of the data. Note that even though both ARM and MFSET cannot model the underlying pickiness characteristics of the datasets, the former does considerably better than the latter. We believe that this is

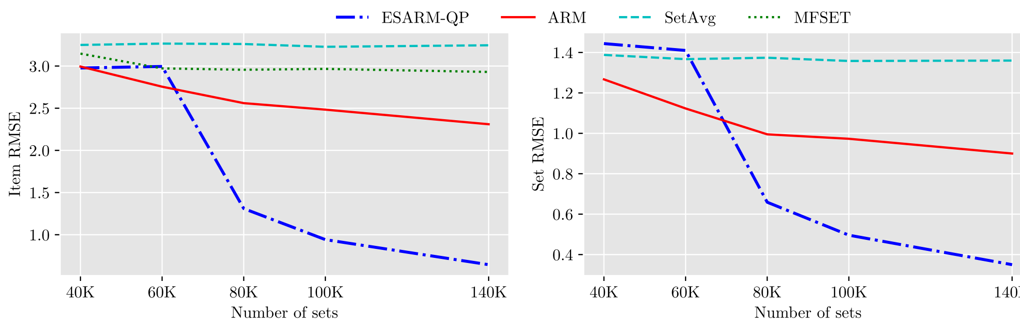

Fig. 9. The average RMSE obtained by the proposed methods on ESARM-based datasets with different number of sets.

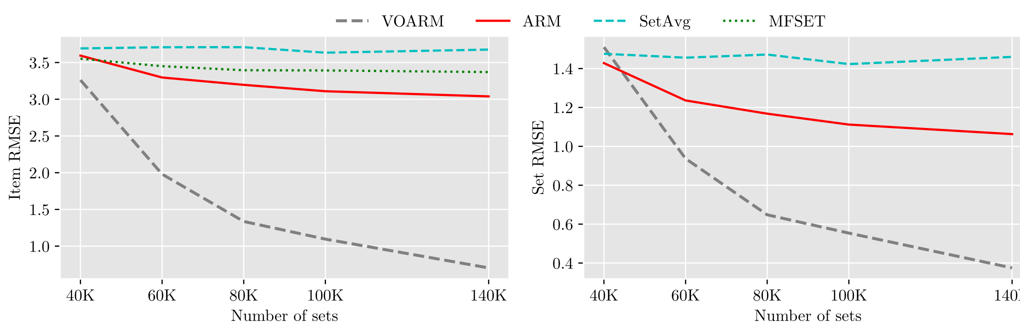

Fig. 10. The average RMSE obtained by the proposed methods on VOARM-based datasets with different number of sets.

due to the fact that ARM's model, which assumes that the average of the set's item-level ratings is equal to that of the set's rating, is significantly more flexible than MFSET's model, which assumes that both the set and all of its items have exactly the same rating. This flexibility allows ARM to better model sets in which there is a high variance among the ratings of the set's items. To test this hypothesis, we performed a series of experiments in which we generated sets with progressively more diversely rated items, which showed that the gap between ARM and MFSET increased with the diversity of ratings in sets (results not shown). Since ARM and MFSET have the same motivation and ARM outperforms the MFSET method, we will present results for the ARM method in the remaining section.

6.2.2 Recovery of Underlying Characteristics. We examined how well ESARM and VOARM recover the known underlying characteristics of the users in the datasets. Figure 11 plots the Pearson correlation coefficient of the actual and the estimated weights that model the users' level of pickiness in VOARM (i.e., β_u parameters). The high values of Pearson correlation coefficients in the figure suggest that VOARM is able to recover the overall characteristics of the underlying data. Additionally, this recovery of underlying characteristics increases with the increase in the number of sets. To investigate how well ESARM can recover the underlying characteristics, we computed the fraction of users for whom the extremal subset having the highest weight (w_{ui}) is same as that of the extremal subset used to generate the rating on sets. Figure 12 shows the percentage of users for whom the extremal subsets are recovered by ESARM. As can be seen in the figure, the fraction of users recovered by ESARM increases significantly with the increase in the number of sets. The better performance of ESARM on the larger datasets suggests that to recover the underlying

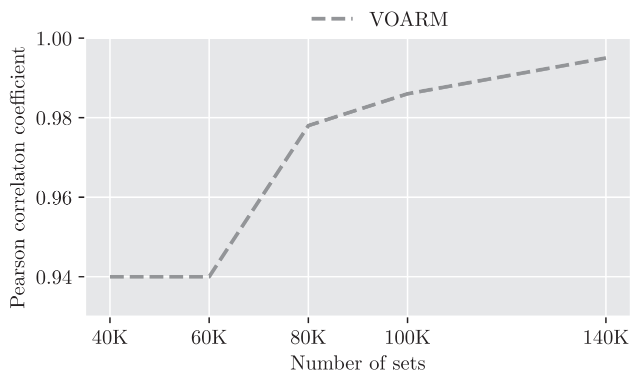

Fig. 11. Pearson correlation coefficients of the actual and the estimated parameters that model a user's level of pickiness in the VOARM model.

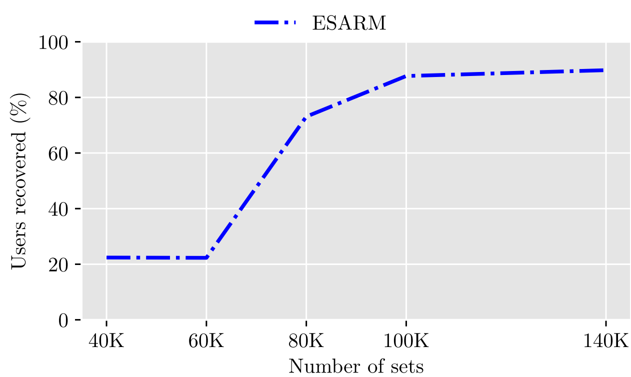

Fig. 12. The percentage of users recovered by ESARM, i.e., the users for whom the original extremal subset had the highest estimated weight under these models.

characteristics of the data accurately, ESARM needs significantly more data than required by the VOARM method. Furthermore, for both ESARM and VOARM methods, we have a low recovery when we have 40K to 60K sets in the dataset, and we believe that this low recovery is because the proposed methods, i.e., ESARM and VOARM, do not have sufficient data to recover the underlying characteristics, and once we increase the number of sets, i.e., ≥80K, we have sufficient data to recover the underlying user-behavior that generated the set-level ratings.

6.2.3 Effect of Adding Item-level Ratings. In most real-world scenarios, in addition to set-level ratings, we will also have available ratings on individual items, e.g., users may provide ratings on music albums and on tracks in the albums. Also, there may exist some users that are not concerned about keeping their item-level ratings private. To assess how well ESARM and VOARM can take advantage of such item-level ratings (when available), we performed three sets of experiments.

In the first experiment, we studied how the availability of additional item-level ratings from the users (U_s) that provided set-level ratings affects the performance of the proposed methods. To this end, we added into the synthetic datasets a set of item-level ratings for the same set of users (U_s) for which we have approximately $100K$ set-level ratings. The number of item-level ratings was kept to $k\%$ of their set-level ratings, where $k \in [5, 75]$, and the items that were added were disjoint from those that were part of the sets that they rated. In the second experiment, we investigated how the availability of item-level ratings from additional users (beyond those that exist in the synthetically

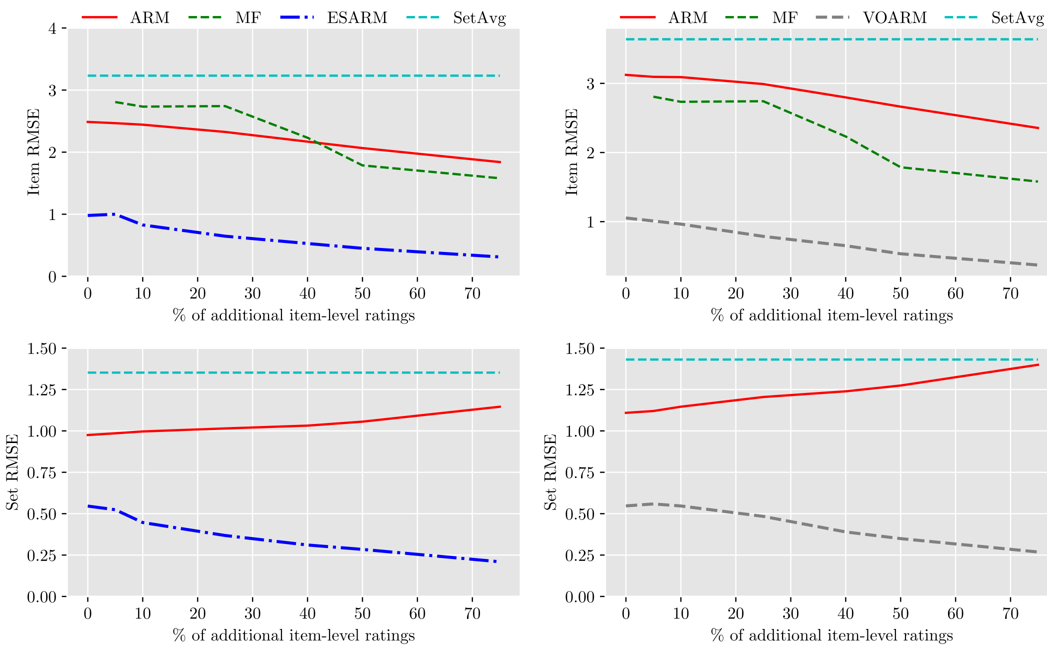

Fig. 13. Effect of adding disjoint item-level ratings for the users in ESARM-based (left) and VOARM-based (right) datasets.

generated datasets) affect the performance of the proposed approaches. We randomly selected 100, 250, and 500 additional users (U_i) and added a random subset of 50 ratings per user from the items that belong to the sets of users in U_s. Figures 13 and 14 show the performance achieved by ESARM and VOARM in these experiments. Additionally, we used the matrix factorization (MF) method to estimate the user and item latent factors by using only the added item-level ratings from the users in U_s. As can be seen from Figure 13, as we continue to add item-level ratings for the users in U_s, there is an increase in accuracy of both the set- and item-level predictions for ESARM and VOARM. Both ESARM and VOARM outperform ARM with the availability of more item-level ratings. For the task of item-level rating prediction, ESARM and VOARM even outperform MF, which is estimated only based on the additional item-level ratings. Figure 14 shows how the performance of the proposed methods changes when item-level ratings are available from users in U_i. Similar to the addition of item-level ratings from users in U_s, ESARM and VOARM outperform ARM with the availability of item-level ratings from users in U_i. The performance of ARM and SetAvg are significantly lower, as both of these methods fail to recover the underlying pickiness characteristics of the dataset and tend to mis-predict many of the item-level ratings. These results imply that using item-level ratings from the users that provided set-level ratings or from another set of users improves the performance of the proposed methods.

In the third experiment, we investigate if using set-level ratings from one set of users (U_s) can improve the item-level predictions for another set of users (U_i) for whom we have only item-level ratings. We randomly selected 500 additional users (U_i) and added a random subset of 50 ratings per user from the items that belong to the sets rated by existing users (U_s). Table 2 shows the performance of item-level predictions for users in U_i and performance on these users after using set-level ratings from users in U_s. As can be seen in the table, the performance of item-level predictions for users in U_i improves significantly after using set-level ratings from existing users

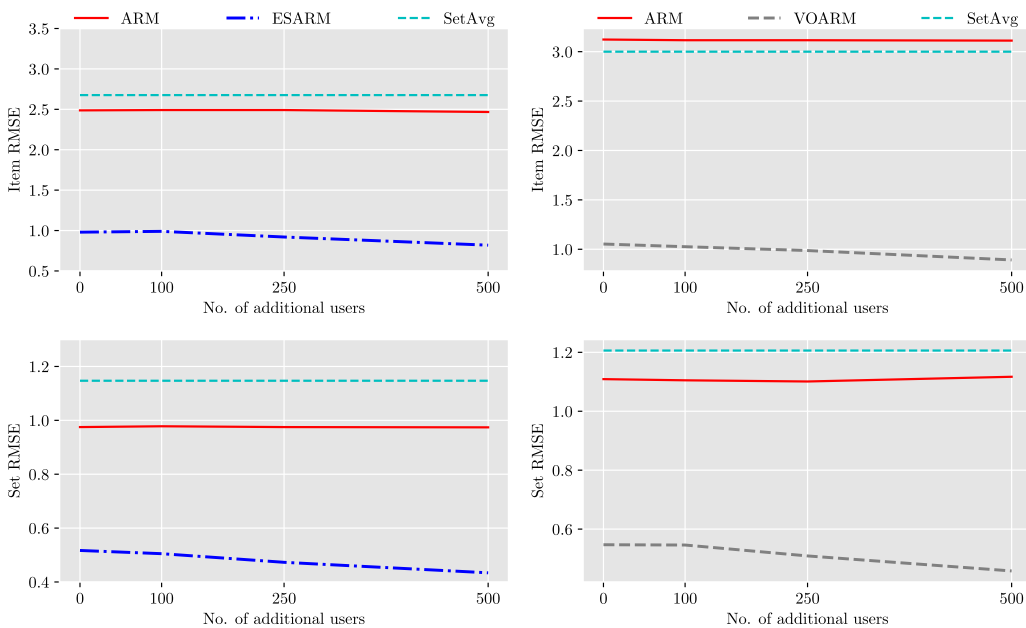

Fig. 14. Effect of adding item-level ratings from additional users in ESARM-based (left) and VOARM-based (right) datasets.

Table 2. Average Item-level RMSE Performance of ESARM and VOARM for a Set of Additional Users (U_i) That Have Provided Only Item-level Ratings

Type of ratings	ESARM	VOARM
Item-level (U_i)	2.860	2.860
Set-level (U_s) + item-level (U_i)	1.811	1.866

U_i is the set of additional 500 users that have provided only item-level ratings. U_s is the set of users that have provided set-level ratings. Item-level (U_i) denotes the item-level ratings of users in U_i. Set-level (U_s) denotes the set-level ratings of users in U_s.

in U_s. These results suggest that using both item- and set-level ratings not only leads to better item recommendations for users (U_s) with set-level ratings but also for those additional users (U_i) who have provided item-level ratings.

6.3 Performance on the Movielens-based Real Dataset

Our final experiment used the proposed approaches (ARM, ESARM, and VOARM) to estimate both set- and item-level rating prediction models using the real set-level rating dataset that we obtained from Movielens users.

6.3.1 Accuracy of Set- and Item-level Predictions. Table 3 shows results for the case when we have only set-level ratings. As can be seen in the table, ARM outperforms the remaining methods for item-level predictions. However, VOARM performs somewhat better than ARM for set-level predictions. The better performance of ARM for item-level predictions is not surprising, as most of the sets in the dataset are rated close to the average of the ratings on items in sets. Also, as seen in our analysis in Section 6.2.2, ESARM needs a large number of sets to accurately recover the

Table 3. The RMSE Performance of the Proposed Methods with User- and Item-biases on ML-RealSets Dataset

Method	Item	Set
SetAvg	0.976	0.630
ARM	0.971	0.624
ESARM	0.979	0.631
VOARM	0.973	0.623

Table 4. Percentage of Item-level Predictions Where Method X Performs Better Than Method Y

Method X \ Method Y	SetAvg	ARM	ESARM	VOARM
SetAvg	—	49.56	53.74	46.41
ARM	50.44	—	51.01	49.85
ESARM	46.26	48.99	—	45.54
VOARM	53.59	50.15	54.46	—

users' extremal subsets. The difference between the predictions of different models was found to be statistically significant (p-value ≤ 0.02 using t-test). Table 4 shows the percentage of the item-level predictions for whom a proposed approach performs better than the other approaches. As can be seen in the table, ARM and VOARM perform better than other methods for the majority of the item-level predictions. In addition, VOARM performs better than ARM for the majority of the item-level predictions. The lower RMSE of ARM for item-level predictions and better performance of VOARM for the majority of the item-level predictions suggest that there are few item-level predictions where VOARM's error is significantly higher than that of ARM. In Section 6.3.3, we will investigate the performance of the proposed methods independently for picky and non-picky users.

6.3.2 Effect of Adding Item-level Ratings. In addition, we assessed how well the proposed methods can take advantage of additional item-level ratings. In the first experiment, we added k% of the users' set-level ratings, where $k \in [10, 75]$, as additional item-level ratings and the items that were added were disjoint from those that were part of the sets that they rated. In the second experiment, we added ratings from 100, 250, and 500 additional users (beyond those that have participated in the survey), and these users have provided on an average 20K ratings for the items that belong to the existing users' sets. In the third experiment, we studied if using set-level ratings from existing users can improve recommendations for additional users that provided only item-level ratings. To this end, we randomly selected 500 additional users and added a random subset of 10 ratings per user from the items that belong to the sets rated by existing users.

Figure 15 shows the results obtained for the first experiment (i.e., adding item-level ratings for the same set of users for which we have set-level ratings). These results show that, with the exception of SetAvg, the performance of all set-based methods improves as item-level ratings are used, and these improvements increase with the percentage of item-level ratings that are used.

Besides the set-based methods, Figure 15 also reports the performance of MF, which uses only the added item-level ratings and the performance of MFOpt, which in addition to the added item-level ratings it uses the actual item-level ratings of the items in the sets (see discussion in Section 5.2.1). These results show that when the number of item-level ratings is small (<30%), MF

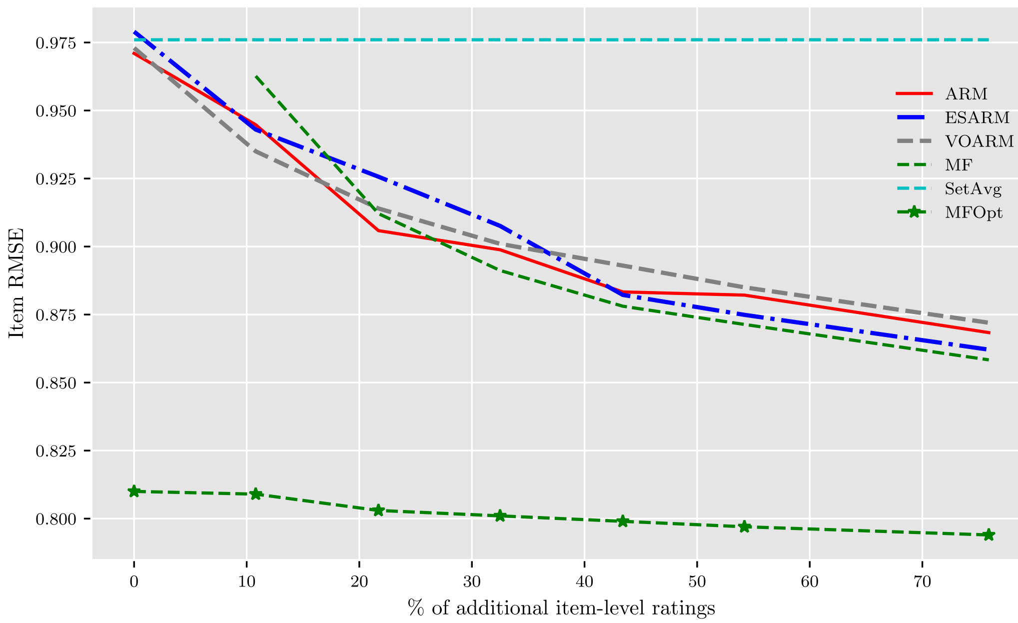

Fig. 15. Effect of adding item-level ratings from the same set of users in the real dataset.

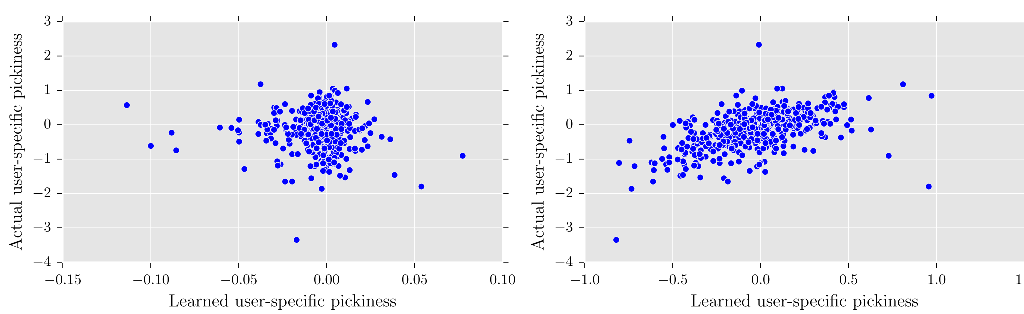

Fig. 16. Scatter plots of the user's original level of pickiness computed from real data and the pickiness estimated by VOARM from set-level ratings (left), and after including 30% of item-level ratings (right).

does not do as well as the set-based methods; however, when there is a sufficiently large number of item-level ratings, MF outperforms the set-based methods, indicating that the set-based methods can improve the recommendations when we have set-level ratings and do not have sufficient item-level ratings to use MF with high accuracy in the recommender system. Additionally, when MF outperforms set-based methods and when set-based methods outperform MF, we found the differences between the predictions from MF and the set-based methods to be statistically significant (p-value < 0.01 using t-test). Figure 16 plots the estimated weights that model a user's level of pickiness in VOARM against the user's level of pickiness, i.e., β_u, computed from the data in Section 6.1. As can be seen in the figure, to some extent VOARM is able to recover the user's level of pickiness after addition of few item-level ratings. Also, the difference between the performance of the proposed methods and MFOpt is reduced as we continue to add more item-level ratings.

In the second experiment, we examined the case when we have item-level ratings from the additional users. In addition to estimating ratings from the proposed methods, we estimated the

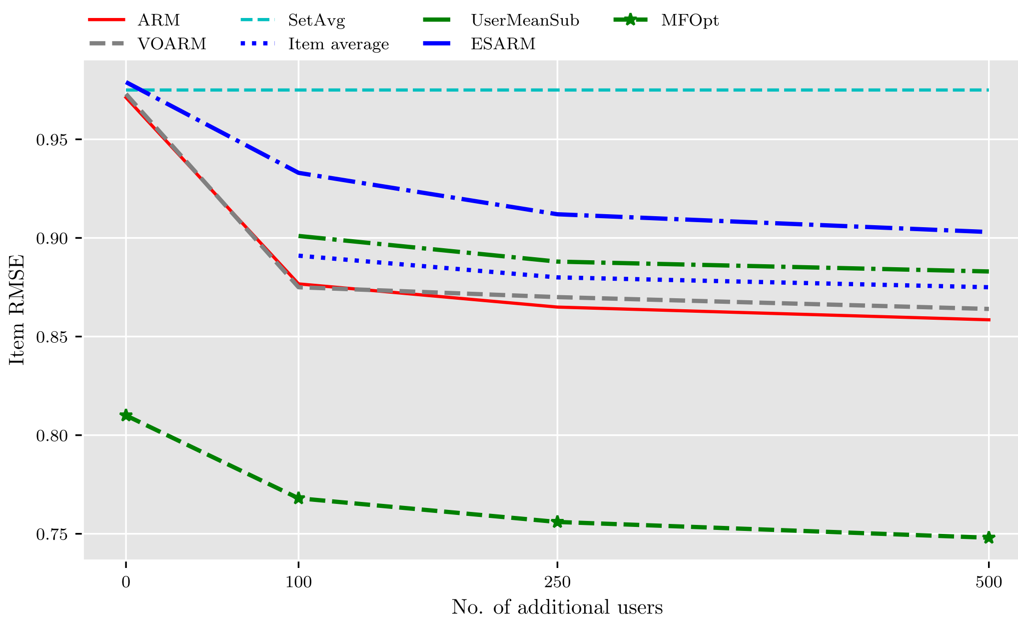

Fig. 17. Effect of modeling biases and adding item-level ratings from disjoint set of users in the real dataset.

Table 5. RMSE for Item-level Predictions for Additional Users That Have Provided Only the Item-level Ratings

Method	Item-level RMSE
MF	1.003
ARM	0.978
ESARM	1.043
VOARM	1.033

ratings at item-level from the two non-personalized methods, i.e., Item average and UserMeanSub, as described in Section 5.2.1. Figure 17 shows the results for these non-personalized methods along with that of the proposed methods. As can be seen in the figure, VOARM and ARM outperform the non-personalized methods, and since ESARM needs a large number of sets to recover the users' extremal subsets, it does not outperform the non-personalized methods. Furthermore, the performance of the proposed methods continue to improve with the availability of more item-level ratings from additional users. Additionally, the difference between the performance of the proposed methods and MFOpt is reduced as we add more item-level ratings from additional users.

In the third experiment, we investigated if using set-level ratings from existing users can improve the item-level predictions for additional users who have provided ratings only at item-level. Table 5 shows the performance of item-level predictions for additional users after using set-level ratings from the existing users and also shows the performance of MF method after using only the additional item-level ratings. As can be seen in the table, ARM outperforms MF for item-level predictions after using set-level ratings from existing users. However, ESARM and VOARM do not perform better than MF for the additional users. Similar to our results on synthetic datasets, it is

Table 6. The Item-level RMSE of the Proposed Methods on Different Subset of Users Using Only Set-level Ratings and After Including Additional Item-level Ratings

	Set only		+Items	
Method	$U_{Non\text{-}picky}$	U_{Picky}	$U_{Non\text{-}picky}$	U_{Picky}
ARM	<u>0.915</u>	1.089	<u>0.879</u>	0.975
ESARM	0.922	1.103	0.898	<u>0.923</u>
VOARM	0.921	<u>1.085</u>	0.892	0.932

The "Set only" column denotes the results of the models that were estimated using only set-level ratings. The "+Items" column shows the results of the models that were estimated using the sets of "Set only" and also some additional ratings on a different set of items from the same users that provided the set-level ratings. U_{Picky} refers to the users who have rated at least 20 sets and have a high level of pickiness, i.e., $|\beta_u| > 0.5$, in real dataset, and $U_{Non\text{-}picky}$ represents the remaining users.

promising that using item-level ratings from the additional users and set-level ratings from the existing users improves the performance not only for the latter, but also for those additional users who have provided only item-level ratings.

6.3.3 Accuracy of Item-level Predictions for Picky Users. Even though ARM performs better than the remaining methods for item-level predictions, we investigated how well ARM, ESARM, and VOARM perform for item-level predictions for the users who have rated at least 20 sets and have a high level of pickiness, i.e., $|\beta_u| > 0.5$. We found 374 users in the dataset that were non-picky ($U_{Non\text{-}picky}$) and 135 users that had a higher level of pickiness (U_{Picky}). Table 6 shows the performance of the proposed methods for item-level predictions using set-level ratings and after including 30% of additional item-level ratings on both U_{Picky} and $U_{Non\text{-}picky}$. As can be seen in the table, for set-level ratings, VOARM performs somewhat better than ARM on U_{Picky}; and after including additional item-level ratings, both ESARM and VOARM outperform ARM on U_{Picky}.

The overall consistency of the results between the synthetically generated and the real dataset suggests that VOARM and to some extent ESARM are able to capture the tendency that some users have to consistently under- or over-rate diverse sets of items.

6.4 Summary

In this work, we investigated two questions related to using set-level ratings in recommender systems. First, how users' item-level ratings relate to the ratings that they provide on a set of items. Second, how collaborative-filtering-based methods can take advantage of such set-level ratings towards making item-level rating predictions. Based on the set of experiments that were presented, we can make the following overall observations:

(1) The ESARM and VOARM set-rating models can explain the ratings provided by users on sets of items better than the ARM model, with ESARM doing slightly better than VOARM (Section 6.1).
(2) The proposed models can use both item- and set-level ratings to improve recommendations not only for users who provided ratings on sets, but also for users with only item-level ratings.
(3) When the ratings follow ESARM and VOARM rating models, the proposed models can recover the underlying characteristics in the data and are resilient to noise in these ratings (Section 6.2.2).

(4) Users with high level of pickiness, i.e., $|\beta_u| > 0.5$, VOARM recovers the underlying characteristics in the data better than ARM; and after including additional item-level ratings, ESARM outperforms both ARM and VOARM in terms of recovery of underlying characteristics in the data (Section 6.3.3).

7 CONCLUSION AND FUTURE WORK

In this work, we studied how users' ratings on sets of items relate to their ratings on the sets' individual items. We collected ratings from active users of Movielens on sets of movies and based on our analysis, we developed collaborative filtering-based models that try to explicitly model the users' behavior in providing the ratings on sets of items. Through extensive experiments on synthetic and real data, we showed that the proposed methods can model the users' behavior as seen in the real data and predict the users' ratings on individual items.

For future work, it will be interesting to study how the performance of the proposed approaches vary with the different number of items in sets and how they perform when instead of having a fixed number of items in sets, the sets contain a varied number of items in sets. Furthermore, one can use ratings on sets of items to generate a ranked list of items by optimizing a ranking loss [28] over the ratings on sets of items. Additionally, in our work, we have used the matrix factorization approach to estimate item-level ratings and, alternatively, we can also use other recently proposed approaches (e.g., deep-learning-based methods [15, 31]) to estimate item-level ratings. Also, the performance of the model could be improved by modeling temporal effects on the ratings and by using side-information such as genres or other movie metadata. Finally, it will be interesting to investigate if, similar to the diversity of ratings in the set, there exist other properties at item- or set-level that can affect a user's ratings on sets of items.

REFERENCES

[1] Gediminas Adomavicius, Jesse Bockstedt, Shawn Curley, and Jingjing Zhang. 2011. Recommender systems, consumer preferences, and anchoring effects. In *Proceedings of the RecSys 2011 Workshop on Human Decision Making in Recommender Systems.* 35–42.

[2] Natalie Aizenberg, Yehuda Koren, and Oren Somekh. 2012. Build your own music recommender by modeling internet radio streams. In *Proceedings of the 21st International Conference on World Wide Web.* ACM, 1–10.

[3] Alejandro Bellogín, Alan Said, and Arjen P. de Vries. 2014. The magic barrier of recommender systems—no magic, just ratings. In *Proceedings of the International Conference on User Modeling, Adaptation, and Personalization.* Springer, 25–36.

[4] Idir Benouaret and Dominique Lenne. 2016. A package recommendation framework for trip planning activities. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys'16).* ACM, New York, NY, 203–206. DOI: https://doi.org/10.1145/2959100.2959183

[5] Dirk Bollen, Mark Graus, and Martijn C. Willemsen. 2012. Remembering the stars?: Effect of time on preference retrieval from memory. In *Proceedings of the 6th ACM Conference on Recommender Systems.* ACM, 217–220.

[6] Léon Bottou. 1998. Online algorithms and stochastic approximations. In *Online Learning and Neural Networks*, David Saad (Ed.). Cambridge University Press, Cambridge, UK. Retrieved from: http://leon.bottou.org/papers/bottou-98x.

[7] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization.* Cambridge University Press.

[8] Shuo Chang, F. Maxwell Harper, and Loren Terveen. 2015. Using groups of items for preference elicitation in recommender systems. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '15).* ACM, New York, NY, 1258–1269. DOI: https://doi.org/10.1145/2675133.2675210

[9] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'12).* ACM, New York, NY, 714–722. DOI: https://doi.org/10.1145/2339530.2339643

[10] Dan Cosley, Shyong K. Lam, Istvan Albert, Joseph A. Konstan, and John Riedl. 2003. Is seeing believing?: How recommender system interfaces affect users' opinions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* ACM, 585–592.

[11] Sarah Darby et al. 2006. The effectiveness of feedback on energy consumption. *Rev. DEFRA Lit. Meter., Bill. Direct Disp.* 486, 2006 (2006).

[12] Jon Froehlich, Eric Larson, Sidhant Gupta, Gabe Cohn, Matthew Reynolds, and Shwetak Patel. 2011. Disaggregated end-use energy sensing for the smart grid. *IEEE Pervas. Comput.* 10, 1 (2011), 28–39.

[13] F. Maxwell Harper and Joseph A. Konstan. 2016. The Movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.* 5, 4 (2016), 19.

[14] George William Hart. 1992. Nonintrusive appliance load monitoring. *Proc. IEEE* 80, 12 (1992), 1870–1891.

[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.

[16] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*. IEEE, 263–272.

[17] Roberto Interdonato, Salvatore Romeo, Andrea Tagarelli, and George Karypis. 2013. A versatile graph-based approach to package recommendation. In *Proceedings of the 25th IEEE International Conference on Tools with Artificial Intelligence*. IEEE, 857–864.

[18] 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*. ACM, 426–434.

[19] Yehuda Koren. 2010. Collaborative filtering with temporal dynamics. *Commun. ACM* 53, 4 (2010), 89–97.

[20] Yehuda Koren, Robert Bell, Chris Volinsky et al. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[21] Q. Liu, E. Chen, H. Xiong, Y. Ge, Z. Li, and X. Wu. 2014. A cocktail approach for travel package recommendation. *IEEE Trans. Knowl. Data Eng.* 26, 2 (Feb. 2014), 278–293. DOI: https://doi.org/10.1109/TKDE.2012.233

[22] Qi Liu, Yong Ge, Zhongmou Li, Enhong Chen, and Hui Xiong. 2011. Personalized travel package recommendation. In *Proceedings of the 11th IEEE International Conference on Data Mining*. IEEE, 407–416.

[23] Yidan Liu, Min Xie, and Laks V. S. Lakshmanan. 2014. Recommending user generated item lists. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys'14)*. ACM, New York, NY, 185–192. DOI: https://doi.org/10.1145/2645710.2645750

[24] Sean M. McNee, Shyong K. Lam, Joseph A. Konstan, and John Riedl. 2003. Interfaces for eliciting new user preferences in recommender systems. In *Proceedings of the International Conference on User Modeling*. Springer, 178–187.

[25] Joshua L. Moore, Shuo Chen, Thorsten Joachims, and Douglas Turnbull. 2012. Learning to embed songs and tags for playlist prediction. In *Proceedings of the International Society for Music Information Retrieval Conference*. 349–354.

[26] Tien T. Nguyen, Daniel Kluver, Ting-Yu Wang, Pik-Mai Hui, Michael D. Ekstrand, Martijn C. Willemsen, and John Riedl. 2013. Rating support interfaces to improve user experience and recommender accuracy. In *Proceedings of the 7th ACM Conference on Recommender Systems*. ACM, 149–156.

[27] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, and John Riedl. 2002. Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of the 7th International Conference on Intelligent User Interfaces (IUI'02)*. ACM, New York, NY, 127–134. DOI: https://doi.org/10.1145/502716.502737

[28] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI'09)*. AUAI Press, 452–461. Retrieved from: http://dl.acm.org/citation.cfm?id=1795114.1795167.

[29] Neil Rubens, Mehdi Elahi, Masashi Sugiyama, and Dain Kaplan. 2015. Active learning in recommender systems. In *Recommender Systems Handbook*. Springer, 809–846.

[30] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW'01)*. ACM, New York, NY, 11.

[31] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 111–112.

[32] Mohit Sharma, F. Maxwell Harper, and George Karypis. 2017. Learning from sets of items in recommender systems. In *Proceedings of the 9th International Conference on Information, Process, and Knowledge Management.*

[33] Pinata Winoto and Tiffany Y. Tang. 2010. The role of user mood in movie recommendations. *Expert Syst. Appl.* 37, 8 (Aug. 2010), 6086–6092. DOI: https://doi.org/10.1016/j.eswa.2010.02.117

[34] Min Xie, Laks V. S. Lakshmanan, and Peter T. Wood. 2010. Breaking out of the box of recommendations: From items to packages. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM, 151–158.

[35] Min Xie, Laks V. S. Lakshmanan, and Peter T. Wood. 2011. Comprec-trip: A composite recommendation system for travel planning. In *Proceedings of the 27th IEEE International Conference on Data Engineering (ICDE'11)*. IEEE, 1352–1355.

Received September 2017; revised December 2018; accepted April 2019