Recommendations with Optimal Combination of Feature-Based and Item-Based Preferences

Mona Nasery
Politecnico di Milano
Milano, Italy
mona.nasery@polimi.it

Matthias Braunhofer
Free University of
Bozen-Bolzano
Bolzano, Italy
mbraunhofer@unibz.it

Francesco Ricci
Free University of
Bozen-Bolzano
Bolzano, Italy
fricci@unibz.it

ABSTRACT

Many recommender systems rely on item ratings to predict users' preferences and generate recommendations. However, users often express preferences by referring to features of the items, e.g., "I like Tarantino's movies". But, it has been shown that user models based on feature preferences may lead to wrong recommendations. In this paper we cope with this issue and we introduce a novel prediction model that generate better item recommendations, especially in cold-start situations, by exploiting both item-based and feature-based preferences. We also show that it is possible to optimize the combination of the two types of preferences when actively requesting them to users.

Keywords

Preference Model; Recommender Systems; Active Learning; Cold-Start

1. INTRODUCTION

Many recommender systems (RSs) compute recommendations exploiting ratings or likes for items, hence user evaluations for items, not for their specific properties. But with the increased availability of on-line information, a number of researches have tried to leverage information about features of items (e.g., in the movie domain, movie genres, cast, etc.) in order to better estimate users' true interests. Despite the large differences between the proposed approaches, which are discussed in the next section, all of them are item-centric, and only ratings or likes over items do describe the user preferences.

However, it is common for users to search for items by features and to express their preferences for items by commenting their properties/attributes. For instance, a user may argue that she likes a movie because of the actors, and may not be interested in, or like, other features of the considered item. Aiming at understanding the relationships between user preferences expressed over items and on their features, in a previous work [12] we collected a dataset of explicitly formulated preferences, in terms of likes of users for both movie features and movies. We discovered that these two types of preferences often do not align and user models based on these two types of information may lead to different recommendations.

So, if these two types of preferences are conflicting can we still effectively leverage both of them? And, more precisely, can the preferences expressed over item features still be useful to predict which items the user will like? In this paper, we address this issue which is detailed in the following two questions:

- Can preferences over features and over items be beneficially combined in order to generate better recommendations?
- If the system has the option to collect selectively any of them, what is the optimal combination of these two types of preferences?

We tackle the first question by proposing a novel matrix factorization method, which is called FPMF (Feature Preferences MF) that incorporates user feature preferences to predict user likes and we compare it with: a) plain matrix factorization, b) most popular recommendations, which are both based only on item "likes", and c) content based filtering, which is based only on feature "likes". In order to address the second question, we consider a feature like or an item like as one single piece of information that could be asked to a user. Hence, we identify the right assortment of preference types when one, two, three, etc. preferences (likes) are available or can be asked to the user. We evaluated the performance of the proposed model with respect to various metrics (accuracy, novelty and coverage). Our results show that the combined model, based on both item likes and feature preferences, is effective and outperforms the compared models, especially in the new-user situation.

2. RELATED WORK

Many recent research works have tried to build recommender systems by exploiting additional information about users or items, i.e., in addition to the user preferences collected in a rating matrix. In [5] the authors proposed a feature-based recommender for situations when there are not enough ratings to measure the similarity between users or items. Their idea is that users who bought items with specific features also buy items with the same/similar features. Ning and Karypis [14] developed four algorithms that

UMAP '16, July 13-17, 2016, Halifax, NS, Canada

DOI: http://dx.doi.org/10.1145/2930238.2930282

incorporate item features for top-N recommender systems and showed that their methods achieve a performance improvement (Hit Rate and Average Reciprocal Hit-Rank) by exploiting side information. Item features information have been also leveraged in latent factor models [8]. Notwithstanding the fact that these approaches do use information about item features to improve the recommender, they differ from our approach as they do not consider signals of user preferences for features, as we do.

Other research works have used user-defined features (e.g., tags) to generate better recommendations. Sen et al. in [17] proposed "Tagommenders", tag-based recommendation algorithms that predict users' ratings for movies based on their inferred tag preferences. Lops et al. in [11] addresses the cold start problem by introducing a tag-based recommender system that combines collaborative filtering with content-based technique. In [3] tags together with ratings data were used in a cross-domain scenario to address the cold start problem. We note that, although tags seems to convey preferences over features, the two are different concepts and tagging an item does not necessarily signal the preference of the user for the attribute denoted by the tag.

On another direction, some researchers have tried to actively elicit user preferences over items [16], or contextual conditions that influence the user preferences [1]. One drawback of eliciting preferences on items is that the user might not know the items that are asked to rate. In this article we show that item and feature preferences can be combined in an optimal mix, thus enabling a new kind of active learning approach so that if the user is not able to reply to the system request to rate an item, she can easily tell the system what features of the proposed item she likes.

3. FEATURE PREFERENCE MF

In order to incorporate known feature preferences of the users in a matrix factorization prediction model, we follow a strategy similar to that used in [2, 9, 4]. Hence, we interpret user u's likes for features as user's attributes and we map them to a set of Boolean attributes $A(u)$ of u. For instance, a user u who likes *action*, and *horror* genre movies as well as movies with actor *sylvester_stallone*, will be considered as possessing the Boolean attributes $A(u) = \{ genre_{action}, genre_{horror}, actor_{sylvester\_stallone} \}$.

Having done that, the resulting set of Boolean attributes $A(u)$ can be taken into account when computing item like predictions by introducing a new additional latent factor vector y_a for each attribute $a \in A(u)$. Accordingly, given a user $u \in U$ and an item $i \in I$, u's like for item i, x_{ui}, is estimated using the following model, which is called Feature Preferences Matrix Factorization (FPMF):

$$\hat{x}_{ui} = (p_u + \sum_{a \in A(u)} y_a)^\top q_i \tag{1}$$

where p_u, y_a and q_i are f-dimensional latent factor vectors corresponding to user u, attribute a and item i respectively. This model is ideal for cold-start situations as it is capable of computing like (or rating) predictions even if no prior likes for the user are available. In this case the latent factor vector p_u is ignored and predictions are calculated solely on the basis of feature preferences.

In this paper, we deal with a positive-only user feedback dataset (likes), which is a very common situation. To efficiently handle this kind of feedback, we adopt the Alternating Least Squares (ALS) optimization technique proposed by Hu et al. [6] and compute the model parameters by minimizing the following cost function:

$$\min_{p*,q*,y*} \sum_{u,i} c_{ui}(x_{ui} - \hat{x}_{ui})^2 + \lambda(\sum_u ||p_u||^2 + \sum_i ||q_i||^2 + \sum_a ||y_a||^2) \tag{2}$$

Here, $x_{ui} = 1$ if user u liked item i, and $x_{ui} = 0$ otherwise. $\hat{x}_{ui}$ is the prediction according to the model in Equation 1. The confidence parameter c_{ui} controls how much the model penalizes mistakes in the prediction of x_{ui}, and is set to $c_{ui} = 1 + \alpha x_{ui}$ as proposed in [6]. The constant α models the increase in confidence for observed feedback. Finally, the regularization parameter λ is used to avoid overfitting the training data.

The model parameters p_u, q_i, y_a are found by minimizing the cost function over all user-item training pairs. To this aim, we extend the ALS-based method with an extra step to compute the additional y_a parameters. ALS is based on the observation that when all parameters except one are fixed, Equation 2 becomes a standard least-squares problem that can be readily computed. The first step is computing the user factors, i.e., we fix the item and user attribute factors, and solve the problem analytically for each p_u, by setting the gradient to zero:

$$p_u = (Q^\top C^u Q + \lambda I)^{-1} Q^\top C^u (x(u) - Q \sum_{a \in A(u)} y_a) \tag{3}$$

where Q is a $|I| \times f$ matrix containing all item factors, C^u is a diagonal $|I| \times |I|$ matrix where $C^u_{ii} = c_{ui}$, and $x(u)$ is a column vector containing all the preferences by u (i.e., the x_{ui} values). Then, we fix the user and user attribute factors, to solve each q_i in a similar fashion:

$$q_i = (Z^\top C^i Z + \lambda I)^{-1} Z^\top C^i x(i) \tag{4}$$

where Z is a $|U| \times f$ matrix containing the vectors $z_u = p_u + \sum_{a \in A(u)} y_a$, C^i is a diagonal $|U| \times |U|$ matrix where $C^i_{uu} = c_{ui}$, and x_i is a column vector containing all the preferences for i. Finally, we fix the user and item factors, and optimize for each y_a:

$$y_a = (Q^\top \sum_{u \in U(a)} C^u Q + \lambda I)^{-1} \sum_{u \in U(a)} Q^\top C^u (x(u) - Q z_{u \setminus a}) \tag{5}$$

where $U(a) = \{ u \in U \mid a \in A(u) \}$ is the set of users with attribute a, and $z_{u \setminus a} = p_u + \sum_{b \in A(u), b \neq a} y_b$ is defined as before but excluding user attribute a. It is important to note that differently from the user and item factors, the user attribute factors depend on the state of all the other attribute factors through the $z_{u \setminus a}$ and thus can not be computed in a parallel fashion.

4. EXPERIMENTAL EVALUATION

4.1 Dataset

We tested our research hypothesis and the FPMF prediction model on a dataset that contains both item and feature likes. The PoliMovie dataset was collected through an online

Table 1: Statistics of PoliMovie dataset

Items / Features	Distinct # of items / features liked	Total # of likes
Movies	1962	4208
Genres	421	2439
Actors	492	1333
Directors	358	890
Movie periods	8	684
Production countries	45	409

survey application which was integrated into a crowdsourcing service called "Microworkers", where 420 users provided their preferences on movies and on various features of them [12]. The preferences in this dataset are positive feedback, i.e., we only have likes on movies and features – not dislikes. Table 1 presents some statistics about the likes we acquired in the PoliMovie survey.

In a previous work, we built two types of user profiles: one based on the user's likes for movies and another based on the user's likes for movie features. We compared them, i.e, the features present in the movies liked by a user versus the features she explicitly liked. The closeness of the two profiles was measured by Jaccard similarity. Our results showed that the two profiles often do not match well [12].

4.2 Evaluation Procedure

In order to address whether preferences over features and over items can be beneficially combined to generate better recommendations, we have measured the performance of our FPMF method on user profiles containing an increasing amount of item preferences (i.e., likes). Then, in a second experiment we tried to identify the best combination of feature and item likes given a fixed amount of available (or askable) preferences of both types.

To achieve this, we conducted a user-based 5-fold cross validation, similar to that proposed in [4, 7]. In particular, we first shuffled the set of users in the whole dataset and split it into five (roughly) equally sized subsets. Then, in each cross-validation iteration, we used all the item and feature likes coming from four merged user subsets as training set to build the prediction model. For each user u in the fifth subset, i.e., the test users, we randomly split her item likes into two subsets: (i) a training set, which is initially empty and then incrementally filled with u's item likes to simulate different numbers of item level preferences, and (ii) a testing set, which is used to compute the performance metrics, namely *Mean Average Precision (MAP)* [10], *Average Popularity* [18] and *Spread* [7], which measure ranking accuracy, novelty and coverage of the recommendations respectively. We also measured *Half-Life Utility (HLU)* [15] and *Mean Percentage Ranking (MPR)* [6], but the results were similar to MAP so we do not report them in Section 5.

Then, in order to reply to the second question we built several models, each one using a fixed number of preferences per user, and we varied the combination of the number of item and feature likes. In this way we were able to measure the effect of different combinations of preference types for a given total number of known preferences.

4.3 Baseline Methods for Evaluation

We have compared the performance of FPMF with the following baseline methods:

- *IMF* computes like predictions using the MF model for implicit feedback datasets proposed by Hu et al. [6] and does not use feature likes. It is known that this method struggles to give accurate predictions in cold-start situations, when little or no information about the users' preferred items is available.
- *CBF* is a pure content-based method that completely ignores item likes and recommends items based on how well their features match the user's features that she likes, based on Jaccard similarity. This method can compute item likes predictions also for users with no item likes history at all.
- *Most Popular* is a non-personalized method which always recommends the most popular items, i.e., those items that received the highest number of likes.

We note that the parameter settings for FPMF and IMF were obtained using the Nelder-Mead optimization method [13], and were as follows: for FPMF, α, f and λ were set to 7.35, 40 and 2.95, whereas for IMF, α, f and λ were set to 15.21, 39 and 13.51.

5. EVALUATION RESULTS

5.1 Evaluation of FPMF Model

Here FPMF uses all the available feature preferences, and the goal is to investigate whether it is beneficial to exploit them along with item preferences. By observing the results in Figure 1 one can note that by exploiting all available feature preferences (i.e., director, cast, country, year and genre preferences) FPMF is able to better rank the recommended items compared to the other models. The improvement of MAP@10 with respect to IMF is larger in the severe cold-start situations: when zero or only a few (i.e., up to 4) users' item likes are available. In particular, when zero item preferences are available, it can be seen that FPMF achieved a MAP@10 of 0.029, which is statistically significantly higher than the MAP@10 of 0.020 and 0.002 achieved by CBF (Wilcoxon signed-rank test, $p = 0.03$) and IMF (Wilcoxon signed-rank test, $p = 0.03$), respectively. Most Popular has the same MAP@10 as FPMF when profile size is zero, but similarly to CBF, which is usable only when no users' item preferences are at disposal, they can not compete with both IMF and FPMF as more and more item preferences become available. It is worth stressing that CBF is based on exactly the same preferences that FPMF exploits to improve IMF, but while in FPMF they are useful to improve the performance achieved by IMF, in CBF they do not seem to bring much information. Apparently this is preference data that cannot be used alone.

In addition to the relevance of the recommended items, we also analyzed the coverage and the popularity of the recommendations produced by the different methods. In Figures 2 and 3 we show the average popularity and the spread of the items recommended by the different methods. We can observe that FPMF and IMF recommend items with similar popularity, except for new users with no or only one item preference. In that case, IMF provides recommendations for items that are less popular – possibly too unpopular – to users, compared to FPMF. The same applies to the CBF method, which recommends the same unpopular items regardless of the number of available item preferences.

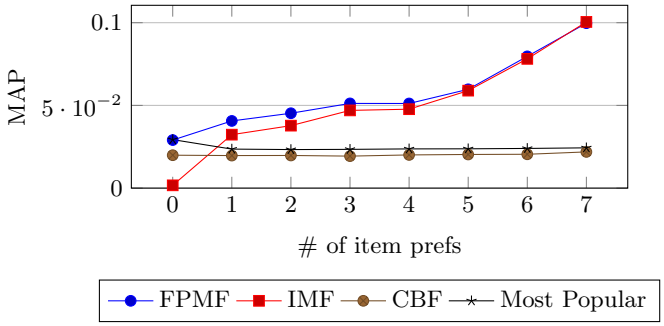

Figure 1: MAP@10 results for different amounts of item preferences/likes

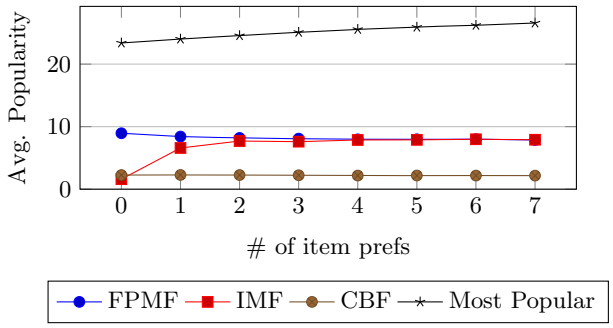

Figure 2: Average Popularity results for different amounts of item preferences/likes

In terms of coverage, the spread of the item distribution is similar among the different methods, except for IMF which recommends the same items over and over to users with zero item preferences.

5.2 Optimal Combination of Item-Based and Feature-Based Preferences

Moving to the second part of our analysis, we considered a feature or item preference as one single piece of information to be asked to the user, and we were interested in finding the best combination of feature-based and item-based preferences. To achieve this, we employed a brute-force feature selection strategy to select the true best combination of information of size k from the given set of N available pieces of information – with k varying from 1 to N. Specifically, for

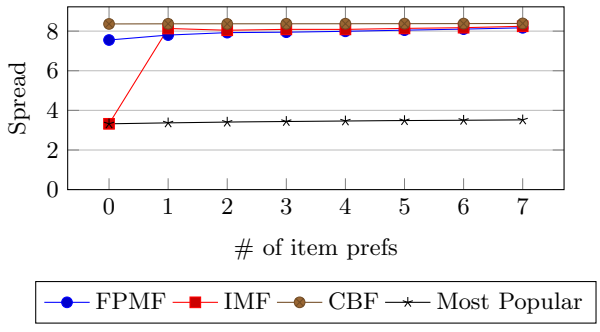

Figure 3: Spread results for different amounts of item preferences/likes

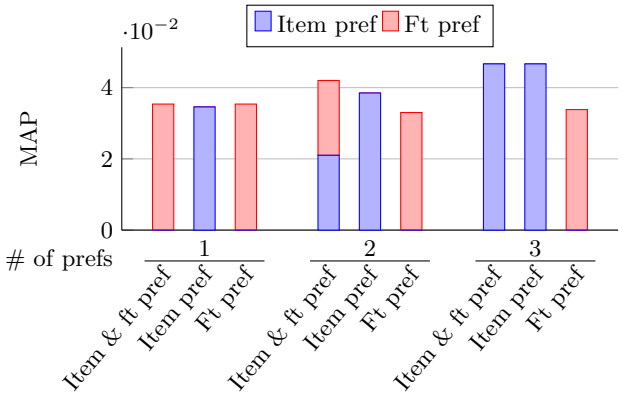

Figure 4: MAP@10 results for different types and amounts of preferences/likes

a given k, we trained FPMF on all possible k-combinations of feature-based and item-based preferences and computed the evaluation metrics on the test set in order to identify the best combination of information.

The results of this analysis are illustrated in Figure 4. It is shown the MAP@10 obtained by FPMF when trained on the best combination of preference information – either feature-based or item-based likes (i.e., "Item & ft pref") – compared to the case when only either item-based likes (i.e., "Item pref") or feature-based likes (i.e., "Ft pref") are used. We note that in the figure the number of preferences (likes) that we considered goes only up to 3 in order to focus on the situations where only a small number of preferences is acquired from the user. In fact, we found that when more than 3 likes can be elicited, the best performance is achieved by considering only item-based preferences.

In conclusion, in the cold start case, in order to achieve the best performance, which type of preferences should be asked to the user must be chosen with care; eliciting only feature preferences can clearly lead to lower MAP@10. Acquiring feature preferences can be useful only when a small number of likes can be requested to the users.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented a model based on both item-based and feature-based preferences (likes) of users, which extends plain MF by incorporating users expressed features preferences to generate better predictions, i.e., better predicting what items the users like. We evaluated the performance of our model in an offline experiment by considering three different metrics: MAP, Average Popularity and Spread. Our main finding is that our model led to more relevant recommendations especially in the severe cold start situation and that can be optimally used in situations where both item-based and feature-based preferences may be available or requested to the user.

As future work, we would like to perform additional experiments on more datasets where both types of preferences are available. Furthermore, we plan to perform an online experiment by conducting a live user study where we could better study the interactions between these two types of preferences and examine the usefulness (performance) of our combined recommendation model.

7. REFERENCES

[1] M. Braunhofer and F. Ricci. Contextual information elicitation in travel recommender systems. In *Information and Communication Technologies in Tourism 2016*, pages 579–592. Springer, 2016.

[2] M. Elahi, M. Braunhofer, F. Ricci, and M. Tkalcic. Personality-based active learning for collaborative filtering recommender systems. In *AI* IA 2013: Advances in Artificial Intelligence*, pages 360–371. Springer, 2013.

[3] M. Enrich, M. Braunhofer, and F. Ricci. Cold-start management with cross-domain collaborative filtering and tags. In *E-Commerce and Web Technologies*, pages 101–112. Springer, 2013.

[4] I. Fernández-Tobías, M. Braunhofer, M. Elahi, F. Ricci, and I. Cantador. Alleviating the new user problem in collaborative filtering by exploiting personality information. *User Modeling and User-Adapted Interaction*, pages 1–35, 2015.

[5] E.-H. S. Han and G. Karypis. Feature-based recommendation system. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 446–452. ACM, 2005.

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

[7] D. Kluver and J. A. Konstan. Evaluating recommender behavior for new users. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 121–128. ACM, 2014.

[8] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[9] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[10] Y. Li, J. Hu, C. Zhai, and Y. Chen. Improving one-class collaborative filtering by incorporating rich user information. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 959–968. ACM, 2010.

[11] P. Lops, M. De Gemmis, G. Semeraro, C. Musto, and F. Narducci. Content-based and collaborative techniques for tag recommendation: an empirical evaluation. *Journal of Intelligent Information Systems*, 40(1):41–61, 2013.

[12] M. Nasery, M. Elahi, and P. Cremonesi. Polimovie: a feature-based dataset for recommender systems. In *ACM RecSys 2015 CrowdRec Workshop*, 2015.

[13] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[14] X. Ning and G. Karypis. Sparse linear methods with side information for top-n recommendations. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 155–162. ACM, 2012.

[15] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 502–511. IEEE, 2008.

[16] N. Rubens, M. Elahi, M. Sugiyama, and D. Kaplan. Active learning in recommender systems. In *Recommender systems handbook*, pages 809–846. Springer, 2015.

[17] S. Sen, J. Vig, and J. Riedl. Tagommenders: connecting users to items through tags. In *Proceedings of the 18th international conference on World Wide Web*, pages 671–680. ACM, 2009.

[18] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM, 2005.