Enhancing New User Cold-Start based on Decision Trees Active Learning by Using Past Warm-Users Predictions

Manuel Pozo*, Raja Chiky, Farid Meziane, Elisabeth Métais

Institut Supérieur d'Eléctronique de Paris, LISITE Lab
28, rue Notre-Dame-des-Champs. 75006 Paris, France
manuel.pozo@isep.fr; raja.chiky@isep.fr; f.meziane@salford.ac.uk; metais@cnam.fr

Abstract. The cold-start is the situation in which the recommender system has no or not enough information about the (new) users/items, i.e. their ratings/feedback; hence, the recommendations are not accurate. Active learning techniques for recommender systems propose to interact with new users by asking them to rate sequentially a few items while the system tries to detect her preferences. This bootstraps recommender systems and alleviate the new user cold-start. Compared to current state of the art, the presented approach takes into account the users' ratings predictions in addition to the available users' ratings. The experimentation shows that our approach achieves better performance in terms of precision and limits the number of questions asked to the users.

Keywords: Active learning for recommender systems; Cold-start problem; New users problem; Decision trees

1 Introduction

The new user cold start is the situation where a recommender system cannot generate personalized recommendations for a new user because it has not learnt yet his preferences. This issue is commonly encountered in collaborative filtering recommendations as they rely mainly on the users' feedback to predict future users' interests [1]. In addition, new users start evaluating the system from their first usage [2]. This is a challenge for both academia and industry because the recommendations' accuracy is directly related to the users' satisfaction and fidelity [3].

The techniques used to alleviate the new user cold start can be categorized into passive learning and active learning. Passive collaborative filtering techniques [1] learn from sporadic users' ratings; hence learning new users preferences is slow [4]. Active techniques interact with the new user in order to retrieve a bunch of ratings that allow them to learn the user's preferences. We focus on active learning techniques for collaborative filtering because they quickly and accurately bootstrap the generation of recommendations for users. In addtion,

* Currently working at *Blackpills*. Email: mp@blackpills.com

collaborative filtering only requires users preferences; analyzing users' ratings from (new) users can achieve better recommendation in cold-start than exploiting other users' attributes (e.g. age, genre) [5].

A naive active learning approach is to question users about their interests and get their answers [6]. Such questions may include: 'Do you like this movie?' with possible answers such as: 'Yes, I do'; 'No, I do not'; 'I have not seen it'. In this context the questions are items and the answers are the users' preferences to these items. However, users are not willing to answer many questions [3, 7]. As a consequence, the main goal is to present short (a maximum of 5-7 questions [2]), but very informative questionnaires. Active learning creates personalized questionnaires which leads to a progressive understanding of the user's preferences. In fact, the personalization of the questionnaires is close to a recommender system concept, although the latter seeks the items the user likes and the former seeks the items the user recognizes.

Our contribution relies on an active learning technique based on decision trees that exploits both available users' ratings and warm users' ratings predictions in order to improve the questionnaire. The experimentation shows that our approach enhances previously suggested ones in terms of accuracy and in a smaller number of questions.

This paper is organized as follows: Section 2 presents the state of the art for active learning using decision trees techniques. Section 3 presents our contribution to enhance active learning based on past warm users' rating predictions. Section 4 shows the experimentations performed and the results of our approach. Finally we conclude and present our future works in Section 5.

2 Related Work

In this research, we focus on active learning techniques in the domain of collaborative filtering recommendations, particularly those using decision trees because: (1) the sequential question paradigm allows a personalization of the questionnaire, and (2) they aim to well profile a new user by posing as less questions as possible. Other techniques, such as Entropy0, Logarithmic Popularity Entropy (LPE) and Harmonic Entropy Logarithmic Frequency (HELF), are not discussed [6]. Specifically, we do not mention passive learning techniques [8–10], content-based techniques [11] or other hybrid techniques [12–14]. For more information about these approaches, the readers can refer to [3, 7, 4].

Recent researches focus on user partitioning techniques that allow to group users of similar tastes into clusters or nodes, and then find out to which group the new user belongs to. In [15] authors use clustering techniques to find the correct users neighbors that match the new user with other users' profiles. This makes it easier to generate recommendations in cold-start. In [2] the authors use non supervised ternary decision trees to model the questionnaire. The decision trees are built off-line to be completely available for new users that receive the questions sequentially. To move to a new question they answer the current one by clicking on one of the three possible answers ('like', 'hate' and 'unknown').

The users' answers lead to a different child node of the trees. This creates a personalized tree path that depends on the past users' answers. On the other hand, this technique uses a collaborative filtering approach to choose questions. Using available users' ratings, they seek the best discriminative item in order to split the population of users into three nodes (users who liked, those who hated and those who do not know this item). The best item is the one which minimizes a statistical error within the users' ratings of the node.

In [16] authors suggest to apply matrix factorization while building every node of a decision tree, yet this is computationally expensive. In [17] the authors proposed to "learn" the active learning technique. They assumed that warm users can be thought of as new users from whom some ratings are known. Thus, this is seen as a supervised decision trees which internally reduces the accuracy of the technique by picking the best discriminative items. Moreover, they split the tree nodes into six, a 1-5 natural scale rating and an unknown node.

The approach in this paper uses decision trees as in [2, 17], and it picks better discriminative items and hence better bootstraps the new users' preferences.

3 Contribution

Supervised and non supervised active learning decision trees aim to find out the best discriminative items for every node of the tree in order to better capture the new users preferences.

Formally, let R be the available ratings. The rating of a user u in an item i is defined by $r_{u,i} \in R$. In addition, let t be a node in the decision trees. We define U_t, I_t, and R_t as the set of users, items and ratings currently in the node t. Furthermore, $R_t(u)$ and $R_t(i)$ are ratings of the user u and item i in the node t. Given the current node t, these techniques iterate over all candidate items $i \in I_t$ by analyzing users' ratings on i. The users populations are then grouped into users' who rated item i and users who did not. Typically, the latter is more populated due to the sparse nature of the available dataset. Furthermore, the users who rated item i can be grouped into further categorizations, e.g. users who liked/hated, or who rated '1, 2, 3, 4, 5'. Then, the population of users in these nodes, and their ratings, are used to evaluate the performance of choosing i as one discriminative item of R_t.

Our contribution exploits the predictions over the existing R. Thus, we define P as the predicted set of R, so that for each $r_{u,i} \in R$ there is a prediction $p_{u,i} \in P$. The set P is computed by using collaborative filtering techniques, e.g matrix factorization. Highlight that the number of users, items, and entries in R and P are the same. Finally, P_t is the set of predictions currently in the node t, and $P_t(u)$ and $P_t(i)$ are the set of users and items predictions in the node t.

Current decision trees techniques exploit only the available ratings in R in order to (1) find the discriminative items, (2) split the users' population, and (3) compute predictions over the candidate items. These techniques use a simple item prediction method based on the "item rating average" in order to evaluate a prediction accuracy and to compute prediction labels for candidate items.

Note that this technique is fast and accurate in large datasets, which allows a quicker generation of predictions from the available ratings in R_t. On the other hand, using more accurate prediction techniques is possible but (1) it can be very expensive and time consuming to do it for every node of the tree, and (2) the predictions needed in decision trees are item-oriented regardless of the user (the same prediction value to any user).

We propose to change this paradigm by using more accurate predictions over the available ratings R. The main idea is to introduce the prediction P as a new source. Hence, R and P are available from the root node of the tree. Then, when the node is split into child nodes, R_t is split into $R_{t-child}$. As long as we want to preserve that for every rating $r_{u,i} \in R_t$ there is an associated prediction $p_{u,i} \in P_t$, for every node t, we split P_t into $P_{t-child}$ as well. In addition, we propose to use the available ratings in R only to split the users population, and P to find out the best discriminative items to enhance the prediction label of candidate items.

This makes sense since finding discriminative items and label predictions are associated with computing an error. As long as P is built by using more accurate methods than the "item rating average", this error is minimized efficiently. We propose using efficient algorithms, such as matrix factorization [18]. The main drawback of using matrix factorization is that it computes different item predictions for different users. The decision trees require a unique item prediction value to be applied to any user. In [2, 17] the authors use the "item rating average" within R_t. We suggest using a similar method, with a major difference that is computing the "item prediction average", which is indeed the average of the predictions within P_t.

In fact, collaborative filtering methods are very accurate for recommending items to users by replicating the users' rating behavior. As a consequence, they are good as well in guessing the average prediction of users, items, and in general the average rating value of the dataset. Figure 3 supports this statement. In addition, this is true as well for the "item prediction average". Figure 2 develops this by considering different group of users split by quantity of ratings. We observe that "item prediction average" based on matrix factorization predictions (MF-Avg) are close to the item rating average predictions (Item-Avg), while as normal the matrix factorization (MF) outperforms these predictions.

3.1 Apply Warm Predictions to decision trees algorithms

The difference between supervised and non-supervised approaches is that the former considers that some users' ratings can be used to validate the technique. As a consequence, these ratings can be used as a validation set to evaluate the accuracy of the tree node. On the contrary, since non supervised techniques do not have any validation, they compute a statistical error based on the available ratings in the node. Nevertheless, in both approaches a validation is not possible in the 'unknown' nodes, since by definition, there is no rating label for these users to this item. As a consequence, a statistical error is mandatory in this case. Our approach uses similar statistics as [2].

Statistic	Movielens	MF
1st Quartile	3.00	3.13
Median	4.00	3.58
Mean	3.51	3.51
3rd Quartile	4.00	3.96

Fig. 1. Statistics for available ratings and matrix factorization (MF) predictions of Movielens 10M [19].

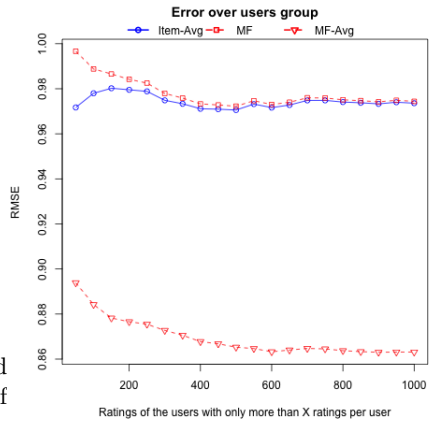


Fig. 2. Prediction techniques and average comparisons regarding the RMSE for Movielens 10M [19].

Non Supervised Approach In [2], the authors define a set of statistics and an internal error using these statistics to find out the best discriminative item. In this approach, the best item is the one which reduces this error. In addition, as long as the tree nodes contain many ratings, they use the item rating average method to compute item label predictions for items.

In our approach we use the same statistics to compute the same error, with two major differences. First, the available ratings are only used to split the population of users. As a consequence, the statistics and the items predictions are computed by using the proposed set of predictions P. Second, once a discriminative item is chosen in a parent node it does not pass to the child nodes. This is done for two reasons: (1) to avoid to choose the same item, and hence, to avoid to pose twice the same question to the same user, and (2) to delete the influence of the items' ratings in the child nodes. In fact, one can avoid choosing an item without deleting their ratings as done in [2]. This approach is described in Algorithm 1.

Supervised Approach In [17], the authors suggested using warm-users as cold-users from whom some interests are known. This assumption allows to create a supervised decision trees where some labels are known for validation purposes.

We suggest again to use the predictions P over the available ratings in R in order to enhance this technique. We make use of R to split the users' population, meanwhile P is used to (1) validate the approach, and (2) obtain items label for the chosen discriminative items. The validation requires the items predictions, which in [17] is given by the item rating average within the child node. As long as P contains the predicted values $p_{u,i}$, this validation is more accurate. In addition, the item prediction average over P is also used to obtain a prediction,

Algorithm 1 Non-supervised decision tree algorithm

```
1: function BuildDecisionTree(R_t, P_t, currentTreeLevel)
2:     for rating r_{u,i} in R_t do
3:         accumulate statistics for i in node t using p_{u,i}
4:     end for
5:     for candidate item j in I_t do
6:         for r_{u,j} in R_t(j) do
7:             obtain P_t(u) and split U_t 3 child nodes based on j
8:             find the child node where u has moved into
9:             for rating p_{u,i} in P_t(u) do
10:                accumulate statistics for i in node t − child using p_{u,i}
11:            end for
12:        end for
13:        derive statistics for j in node tU from the tL and tD statistics
14:        candidate error: e_t(j) = e_{tL}(j) + e_{tD}(j) + e_{tU}(j)
15:    end for
16:    candidate item i* = argmin_i
17:    compute p_{i*} by using item prediction average
18:    if currentTreeLevel ¡ maxTreeLevel then
19:        create 3 child nodes U_{t−child} based on i* ratings
20:        for child in child nodes do
21:            exclude i* from R_{t−child}
22:            BuildDecisionTree( R_{t−child}, P_{t−child}, currentTreeLevel +1 )
23:        end for
24:    end if
25:    return i*
26: end function
```

and we split the nodes into 3 child nodes ('like', 'dislike', 'unknown') rather than 6. Algorithm 2 shows this approach.

3.2 Complexity of the algorithm

The complexity of our approaches for non-supervised decision trees and supervised decision trees is very similar to [2, 17]. In fact, these algorithms follow a similar procedure. The complexity of splitting the users in node t is $O(\sum_{u \in U_t} |R_t(u)|^2)$, and thus, for all the nodes in the same level we use $O(\sum_{u \in U} |R(u)|^2)$. As a consequence, the complexity to build a tree of N questions is $O(N \sum_{u \in U} |R(u)|^2)$. In fact, adding the prediction set P does not affect the complexity of the algorithms, although, the memory footprint of the approaches may vary according to their implementations. Considering that rating and prediction datasets are coded equally, our approach consumes double of the memory size to store the set P.

4 Experimentation

The goal of our experimentation is two-folds (i) to present the behaviour of current techniques in smaller datasets and (ii) to show the performance of our

Algorithm 2 Supervised decision tree algorithm

function BUILDDECISIONTREE(U_t, $R_{t-train}$, $R_{t-validation}$, P_t, *currentTreeLevel*)
  for user $u \in U_t$ **do**
    compute $RMSE_u^1$ on $R_{t-validation}(u)$ and $P_t(u)$
  end for
  for candidate item j from $R_{t-train}$ **do**
    split U_t 3 child nodes based on j
    for user $u \in U_t$ **do**
      find the child node where u has moved into
      compute $RMSE_u^2$ on $R_{t-validation}(u)$ and $P_t(u)$
      $\triangle_{u,i} = RMSE_u^1 - RMSE_u^2$
    end for
  end for
  δ = aggregate all $\triangle_{u,i}$; and pick candidate item $i^* = argmax_i \delta_i$
  compute p_{i^*} by using item prediction average
  if currentTreeLevel ¡ maxTreeLevel and $\triangle_{i^*} \geq 0$ **then**
    create 3 child nodes $U_{t-child}$ based on based on i^* ratings
    for *child* in child nodes **do**
      exclude i^* from $R_{t-child}$
      BuildDecisionTree($U_{t-child}$, $R_{t-child-train}$, $R_{t-child-validation}$, $P_{t-child}$, *currentTreeLevel +1*)
    end for
  end if
  return i^*
end function

presented approach. Recent techniques have presented their results using Netflix dataset. However, this dataset is no longer available for research. Hence, we use the Movielens 10M dataset [19], which contains 71567 users, 10681 items and 10 million ratings. Since our approach considers external techniques prediction as a new source, in order to build our decision trees we use matrix factorization [18] due to its accuracy. We compare our approach in non supervised decision trees, as in [2], and in supervised decision trees, as in [17].

In order to compare the approaches we use the RMSE metric oriented to users, which measures the squared difference between the real ratings and the predicted ratings:

$$RMSE_u = \sqrt{\frac{1}{N}\sum(r_{u,i} - p_i)^2} \tag{1}$$

Where N is the number of ratings of the user u, p_i is the predicted label value of the candidate item in the question node and $r_{u,i}$ is the real rating of the user u for the item i. Hence, the evaluation of the error in one question is the average of the users error in this question number. As a consequence, for this metric the lower is the better.

The experimentation carried out in [2] splits the datasets into 90% training set, D_{train} and 10% test set, D_{test}. However, this is not a real cold-start context

since the same user may appear in both training and test set. We suggest a real cold-start situation. We split the set of users in the datasets into 90% training set, U_{train} and 10% test set, U_{test}. Hence, the users in the training set help to build the decision trees and the users in the test set are considered as new user to evaluate the performance of the approach.

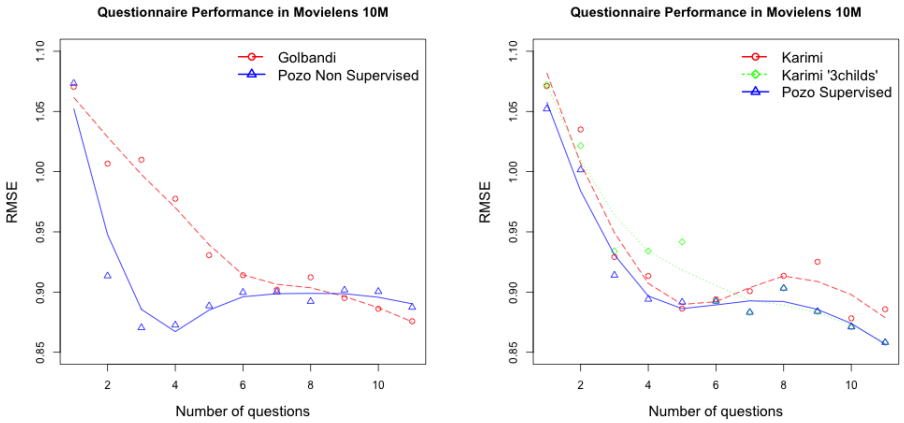
The process we have followed to run this experimentation is as follows. First we split the dataset into U_{train} and U_{test}. Second, we compute the collaborative filtering algorithms over ratings R in U_{train} and we extract the associated predictions P. Third, we train the approach of Golbandi by using U_{train}. Our approach is trained by using both ratings in training set R and the prediction of the training set P. Finally, the performance of the decision trees is evaluated by using the test set U_{test}. The users in this set are used to answer the questions. If the item is known, we compute the RMSE associated to this answer and this question. Then, the user answer a new question. At the end, we compute the average of the accumulated nodes RMSE.

Knowing that the experimentation may depend on the split of the dataset, we run it 50 times and then used the mean value of the RMSE. We use this process to evaluate the performance for the MovieLens 1M and MovieLens 10M. Figure 3(a) shows the results (the mean values and tendency curves) of this experimentation for both MovieLens datasets, where 'Golbandi' is the approach used in [2]. On the one hand, our approach achieves a much lower error in less number of questions. This matches with the needs of active learning; short but very informative questionnaires. This is possible due to the higher accuracy of the matrix factorization. On the the other hand, all the approaches tend to converge into a pseudo-asymptotic behavior. This is due to the fact that nodes in the bottom of the tree (nodes in 8th question) are less populated by users and thus predictions and profiles are less accurate. This particularity is not shown in [2, 17] due to their very large dataset.

We perform a similar experiementation to compare our approach to [17]. This time the authors use a 4-fold set to evaluate their dataset: D_{train} set which is split into U_{train} and $U_{validation}$ sets, and D_{test} set which is split into U_{test} and U_{answer} sets. U_{train} and $U_{validation}$ are to train and validate the evolution of the algorithm. U_{test} and U_{answer} are used to evaluate the performance of the tree at the question q and to answer to that questions. As long as the validation phase aims to optimize the RMSE, the accuracy prediction of the matrix factorization enhances this metric. This yields to better questions and hence the accuracy of the decision tree is enhanced as well. Figures 3(b) shows better results than [17] as well. Further analysis are not described in this paper due to a lack of space.

5 Conclusions and Future Work

The personalization of the active learning technique is crucial to better learn the new users preferences and decision trees are interesting techniques to model questionnaires. Indeed, decision trees can predict the items that new users have already used, although we consider that recent approaches do not correctly ex-



(a) MovieLens 10M dataset. (b) MovieLens 10M dataset.

Fig. 3. Questionnaire performance in RMSE.

ploit the "prediction" inside the decision trees since they use very simple approaches (e.g. item rating average) to make it tractable.

The main idea of our contribution is to train an accurate collaborative filtering techniques with a ratings dataset to generate a prediction dataset. Then, both ratings and predictions dataset are used inside the decision trees. The former properly split the users' population while building the tree. The latter enhances the seek of the best discriminative items (questions) and better predict the associated labels. We have tested this approach in non supervised decision trees and supervised decision trees. The experimentation shows that our approach find better questions to present to users in order to better understand his preferences.

Our future work focuses on (1) detecting the new users preferences directly on the fly, and (2) using new techniques to exploit the information coming from questionnaires. We especially believe that the time (new) users spent to answer a question is very significant for the answer itself. Thus, we focus on "time-aware" recommendation techniques and decision trees to retrieve and exploit not only the users' answers but also the users' behavior.

Acknowledgments

This work has been supported by FIORA project, and funded by "DGCIS" and "Conseil Regional de l'Île de France".

References

1. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Advances in artificial intelligence **2009** (2009) 4

2. Golbandi, N., Koren, Y., Lempel, R.: Adaptive bootstrapping of recommender systems using decision trees. In: Proceedings of the fourth ACM international conference on Web search and data mining, ACM (2011) 595–604
3. Rubens, N., Kaplan, D., Sugiyama, M.: Active learning in recommender systems. In: Recommender Systems Handbook. Springer (2011) 735–767
4. Karimi, R., Freudenthaler, C., Nanopoulos, A., Schmidt-Thieme, L.: Comparing prediction models for active learning in recommender systems. In: Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, and FGDB. October 2015, http://ceur-ws.org (2015)
5. Pilászy, I., Tikk, D.: Recommending new movies: even a few ratings are more valuable than metadata. In: Proceedings of the third ACM conference on Recommender systems, ACM (2009) 93–100
6. Rashid, A.M., Albert, I., Cosley, D., Lam, S.K., McNee, S.M., Konstan, J.A., Riedl, J.: Getting to know you: learning new user preferences in recommender systems. In: Proceedings of the 7th international conference on Intelligent user interfaces, ACM (2002) 127–134
7. Elahi, M., Ricci, F., Rubens, N.: Active learning in collaborative filtering recommender systems. In: E-Commerce and Web Technologies. Springer (2014) 113–124
8. Hofmann, T.: Collaborative filtering via gaussian probabilistic latent semantic analysis. In: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, ACM (2003) 259–266
9. Lemire, D., Maclachlan, A.: Slope one predictors for online rating-based collaborative filtering. In: SDM. Volume 5., SIAM (2005) 1–5
10. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8) (2009) 30–37
11. Peis, E., del Castillo, J.M., Delgado-López, J.: Semantic recommender systems. analysis of the state of the topic. Hipertext. net **6** (2008) 1–5
12. Ziegler, C.N., Lausen, G., Schmidt-Thieme, L.: Taxonomy-driven computation of product recommendations. In: Proceedings of the thirteenth ACM international conference on Information and knowledge management, ACM (2004) 406–415
13. Vozalis, M.G., Margaritis, K.G.: Using svd and demographic data for the enhancement of generalized collaborative filtering. Information Sciences **177**(15) (2007) 3017–3037
14. Barjasteh, I., Forsati, R., Masrour, F., Esfahanian, A.H., Radha, H.: Cold-start item and user recommendation with decoupled completion and transduction. In: Proceedings of the 9th ACM Conference on Recommender Systems, ACM (2015) 91–98
15. Rashid, A.M., Karypis, G., Riedl, J.: Learning preferences of new users in recommender systems: an information theoretic approach. ACM SIGKDD Explorations Newsletter **10**(2) (2008) 90–100
16. Zhou, K., Yang, S.H., Zha, H.: Functional matrix factorizations for cold-start recommendation. In: Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, ACM (2011) 315–324
17. Karimi, R., Nanopoulos, A., Schmidt-Thieme, L.: A supervised active learning framework for recommender systems based on decision trees. User Modeling and User-Adapted Interaction **25**(1) (2015) 39–64
18. Zhou, Y., Wilkinson, D., Schreiber, R., Pan, R.: Large-scale parallel collaborative filtering for the netflix prize. In: Algorithmic Aspects in Information and Management. Springer (2008) 337–348
19. Harper, F.M., Konstan, J.A.: The movielens datasets: History and context. ACM Trans. Interact. Intell. Syst. **5**(4) (December 2015) 19:1–19:19