Incremental collaborative filtering for binary ratings

Catarina Miranda and Alípio M. Jorge
LIAAD - INESC Porto L.A., Portugal
FEP, University of Porto

Abstract

The use of collaborative filtering (CF) recommenders on the Web is typically done in environments where data is constantly flowing. In this paper we propose an incremental version of item-based CF for implicit binary ratings, and compare it with a non-incremental one, as well as with an incremental user-based approach. We also study the usage of sparse matrices in these algorithms. We observe that recall and precision tend to improve when we continuously add information to the recommender model, and that the time spent for recommendation does not degrade. Time for updating the similarity matrix is relatively low and motivates the use of the item-based incremental approach.

1. Introduction

Recommender systems [4] provide advice to users about items they might wish to purchase or examine. The best known type of recommender system is Collaborative Filtering (CF) [4], where items are recommended to a user based on values assigned by other people with similar taste [7].

In a Web environment, recommender systems can be used to facilitate navigation and, given its dynamics (new users appearing, new items, and preferences changing), recommender models must be constantly updated. One possible approach is to refresh periodically the model/matrix, to reflect the changes which has the disadvantage of having to rebuild the whole model every time and it does not use the latest information for recommendations. The alternative is an incremental approach - every time a new session occurs, the model is changed accordingly.

Recommender systems can be item-based or user-based [6]. Either way, CF models are built on the basis of sets of ratings given by users to certain items. In the realm of recommender systems for the Web, the only available information is whether a given user has accessed this or that item (e.g. web page). These data are processed as binary implicit ratings. In this paper we are dealing only with this kind of data. Papagelis et al. [2] present an incremental user-based CF method for continuous explicit ratings, based on incremental updates of user-user similarities.

In this paper we propose an item-based incremental algorithm for binary implicit ratings. We empirically evaluate non-incremental and incremental item-based algorithms as well as an incremental user-based algorithm, both in terms of time spent and predictive performance. We also measure the effect of the use of sparse matrices.

2. Collaborative Filtering

CF works by collecting a database of ratings for items by users. Here, we assume that these ratings are binary, 1 if the user has seen the item, 0 otherwise. We use this information as a proxy for preference. Thus, given a new session, a set of pairs <user, item> for the same user u, to a CF algorithm, the aim is to recommend items to u. For that, the new session is directly or indirectly matched against the historical database D to obtain items that are likely to be preferred by u.

The user-based approach proceeds by looking for N users in D that are most similar to u. The items preferred by these neighbors are the recommendations to make. In the item-based approach we look for N items similar to the items in the session. This subtle difference carries important computational consequences [6]. Similarity between users/items can be defined in different ways, in this work we have adopted the cosine measure to construct the similarity matrix S.

Computational issues: Whereas in the user-based approach, recommendations for a new session are generated by analyzing the whole database D, in the item-based approach we only need the similarities between each pair of
items. Since typically the number of items is orders of magnitude smaller than the number of users, this results in an important memory and computational reduction [6].

**Incremental CF:** To reduce the above described problem, [2] presents an incremental user-based CF method which incrementally updates the user-user similarity matrix. In this paper we propose an incremental item-based CF approach for binary ratings. Thus, we benefit of the incrementality facility, maintaining fresher models (matrices) as much as possible, and of the computational advantages of the item-based approach. With the non-incremental approach the similarity matrix is reconstructed periodically from scratch. In the incremental case, the similarity matrix is updated after each new session.

### 3. Algorithms

Here we present the algorithms under study. All of them start with a database $D$ of pairs $<u, i>$ (user $u$ saw the item $i$). All the algorithms start by computing the similarity between pairs of items or users (similarity matrix $S$). All take parameters $Neib$ (number of neighbors to be found for a given user/item) and $N$ (number of recommendations to be made). All the algorithms maintain a set of known $Users$ and $Items$. The active user is $u_a$.

**Algorithm A1:** This is the baseline non-incremental item-based recommender.

- Determine the activation weight of each item never seen by $u_a$
- Recommend to $u_a$ the $N$ items with higher activation weight

The activation weight of $i$ is $W(i) = \sum_{\text{items in the neib seen by } u_a} S[i, .]/\sum_{\text{items in the neib}} S[i, .]$.  

**Algorithm A2:** Similar to A1 but using data structures that spend less memory when storing sparse matrices.

**Algorithm A3:** Incremental version of A2. Besides $S$, we also save in memory the matrix $Int$ (necessary for updating $S$) with the number of users that evaluate each pair of items.

- Determine the activation weight of each item never seen by $u_a$
- Recommend the $N$ items with highest activation weight and update $S$ and $Int$

Let $I$ be the set of items that $u_a$ evaluated in the active session. The updating of $S$ and $Int$ is done as follows:

- Add $u_a$ to $Users$, new items in $I$ to $Items$ and, for each new item, a row and column to $Int$ and $S$
- For each pair of items $(i, j)$ in $I$, $Int_{i,j} = Int_{i,j} + 1$
- For each item $i_a$ in $I$ update the corresponding row (column) of $S$ using $Int$

**Algorithm A4:** Incremental user-based algorithm [2]. Also uses sparse matrices. Besides $S$ we also keep in the matrix $Int.u$ (needed to update $S$), with the number of items evaluated by each pair of users, and the database $D$.

- Update $Int.u$ and $S$
- Determine the activation weight of each item never seen by $u_a$ and recommend the $N$ items with highest activation weight

Let $I$ be the set of items in the active session. The updating of $Int.u$ and $S$ is done as follows:

- Add $u_a$ to $Users$ and the session to $D$
- If $u_a$ is a new user, a row and a column are added to $Int.u$ and $S$
- The row and column of $Int.u$ corresponding to $u_a$ are updated using the new $D$
- Add new items in $I$ to $Items$
- Update the row(column) of $S$ corresponding to $u_a$ using $Int.u$

The activation weight of an item for the user-based algorithm is $W(i) = \sum_{\text{users near } u_a \text{ who saw } \sum_{\text{all the users near } u_a} S[u_a, .]} S[i, .]$.  

**Implementation:** All the algorithms were implemented in R [3]. For sparse matrices we have tried 3 available R packages: spam, SparseM and Matrix. We only show results for (spam), which had the shortest processing time.

### 4. Experimental Evaluation and Results

The aim of our experiments is to assess both the computational and the predictive performance of the 4 algorithms: the impact of incrementality, user-based vs. item-based and the use of sparse matrices. We have used, from our projects, 3 datasets with web accesses from 2 sources: a portal for professionals with interest in economics and management (PE) and a basic e-learning site of a university course (ZP). From the PE data we have built 2 smaller datasets with different proportions of users and items. The ZP data has 509 users/sessions and 295 items, after clipping sessions with less than one access and more than 20 (see Table 1).

<table>
<thead>
<tr>
<th>Data set</th>
<th># Users</th>
<th># Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZP</td>
<td>509</td>
<td>295</td>
</tr>
<tr>
<td>PE200</td>
<td>200</td>
<td>199</td>
</tr>
<tr>
<td>PE802</td>
<td>802</td>
<td>100</td>
</tr>
</tbody>
</table>

**Methodology of evaluation:** We separate the database in training, test and hidden set following the all-but-one protocol [1]. The splits used were 0.8 and 0.4. From each
test session one item is hidden. Testing is done by comparing produced recommendations with the hidden items. The number of neighbors $N_{eib}$ used in CF was 5.

**Measures:** To compare the predictive abilities of the algorithms we have used the measures of Recall and Precision. Let $hits$ be the set with the hits of the recommendation system; $hidden$ be the set with the pairs $<\text{user, item}>$ that are hidden; and $recs$ the set with all the recommendations made by the recommender system. $Recall = \frac{\#hits}{\#recs}$, measures the ability to recommend everything that is relevant to the user. $Precision = \frac{\#hits}{\#hits + \#false}$, measures the ability to recommend only what is relevant, leaving out what will probably not be seen. We also measure the relative $Time$, with respect to algorithm A1, needed to construct $S$ in each method, the average wall time of recommendation per user and to update $S$ and other supporting data structures (per user).

**Time:** We show results (fig. 1) for the recommendation and update/construction time. We can see that the use of sparse matrices has a high cost in terms of recommendation time. A2 is 10 to 45 times slower than A1 when recommending. This is a necessary price to pay, due to the limitation of full matrices. From the comparison of incremental (A3) and non-incremental (A2) item-based algorithms, we can see that the recommendation time is very similar. A2 tends to be worse, but for spurious reasons. More importantly, the update time in A3 is much lower than construction time in A2, which means that it is much cheaper to update the model than to rebuild it from scratch every time we have new information. The exception is the ZP data, for the 0.4 split. The exact reasons for this exception are not known. A4 tends to be much slower when recommending, except for the dataset PE200, when the number of users is identical to the number of items (which is not realistic). Update times of A3 and A4 are similar when the number of users is not much larger than the number of items. However, when we have proportionally more users, the update time of A4 degrades considerably.

**Predictive ability:** Figure 2 shows the curves of Recall and Precision for increasing values of $N$ (maximum number of recommendations allowed to the algorithms). We do not show the results for the split 0.4 because they are very similar to 0.8. We can see that the user-based approach tends to have worse results. The item-based algorithms have similar performance in both measures. A1 and A2 give exactly the same results since they differ only in the data structures for storing the matrices.

## 5. Conclusions and Future Work

In this paper we presented an item-based CF algorithm for data with binary ratings. We have compared it with a non-incremental item-based approach and with an incremental user-based approach. The results on 3 datasets have shown that item-based incrementality pays off in terms of computational cost, with low model update times, and that recommendation times are identical to the non-incremental version. The incremental item-based approach had better results than its user-based counter-part, both in terms of time and predictive accuracy. The update of similarities between items is faster than between users because it doesn’t need to revisit the usage database. We have also used sparse matrices to cope with scalability, and tried 3 different R packages. We conclude that the package `spam` yields faster R programs and that there is a relatively high price for using sparse matrices, as expected.

As future work we need to investigate the behavior of the algorithms with larger datasets and in a continuous setting. The incremental algorithm will be included in an adaptive web platform and the impact of incrementality in the accurate response of CF will be studied on a live setting.

**Acknowledgments:** Fund. Ciência e Tecnologia POSC/EIA/ 58367/2004/ Site-o-Matic and PTDC/EIA/81178/2006 Rank! Projects co-financed by FEDER.

**References**


Figure 1. Recommendation and Update Time of each algorithm using the relation to A1

Figure 2. Recall and Precision vs. Number of Recommendations for each used dataset using Split=0.8