Clustering for Collaborative Filtering Applications

Arnd Kohrs – Bernard Merialdo
Institut EURECOM – Department of Multimedia Communications
BP 193 – 06904 Sophia-Antipolis – France
{kohrs,merialdo}@eurecom.fr

Abstract. Collaborative filtering systems assist users to identify items of interest by providing predictions based on ratings of other users. The quality of the predictions depends strongly on the amount of available ratings and collaborative filtering algorithms perform poorly when only few ratings are available. In this paper we identify two important situations with sparse ratings: Bootstrapping a collaborative filtering system with few users and providing recommendations for new users, who rated only few items. Further, we present a novel algorithm for collaborative filtering, based on hierarchical clustering, which tries to balance robustness and accuracy of predictions, and experimentally show that it is especially efficient in dealing with the previous situations.

Keywords: Collaborative filtering, hierarchical clustering, recommender systems, social filtering

1 Introduction

Collaborative filtering (CF) is a recent technique for recommendation and filtering purposes in various applications. Collaborative filtering applications (CF-applications) use databases of user ratings (CF-databases) and algorithms (CF-algorithms) to predict what users might like or dislike. Recent research has concentrated on the predictive quality of CF-algorithms for various domains of filtered items, such as Usenet news, films, music, etc [1, 5] and has focused on CF-algorithms and variations of CF-algorithms.

Typically, CF-algorithms are evaluated using fully-grown CF-databases. In this paper, we focus on problematic states of CF-databases, which are especially interesting for providers of CF-services. Initially, CF-databases do not contain sufficient information for the CF-algorithms to provide good recommendations. The providers of CF-services have to promote their applications so that enough clients participate. Similar lack of information also occurs in a fully-grown CF-database when a new client starts using a CF-application. This is due to the fact that the quality of the predictions for a user relies strongly on the amount of ratings that he has so far provided.

Due to the fact, that currently used CF-algorithms do not specifically address lack of data in CF-databases, we first present a new colloaborative filtering algorithm in section 2, which is based on hierarchical clustering and tries to combine robustness and accuracy, especially when little data is available.

In section 3 we then identify two general cases which correspond to problematic states of a CF-database, the *Bootstrap-Case* and the *New-User-Case*. Through simulations, we prove that for these cases our algorithm performs better than classically used algorithms.

2 Using Clustering for Collaborative Filtering

CF-databases are sets of ratings. Each rating expresses, on a numerical scale, how much a user has valued a certain item. Ratings can be represented by the sparse matrix $score(user, item)$.

Several algorithms are classically used for collaborative filtering. In particular, we compare an algorithm using Pearson correlation [3, 5, 1, 2] (Pearson algorithm for short), the Mean-Squared-Differences algorithm (MSD algorithm) [5], and as a worst-case, an algorithm, which uses the average ratings of each item, without adaption to a specific user as predictions (Base algorithm). The algorithms vary in the way similarity is determined and in the way the ratings are accumulated to form predictions.

In the literature sparsity of ratings has been discovered as a major problem for CF-systems[4]. Several approaches have been made to reduce the sparsity by creating artificial or implicit ratings[4, 3]. We address the sparsity of ratings by a new CF-algorithm, which takes most advantage of the given data. We, therefore, propose clustering of rating data to overcome sparsity. Our algorithm is based on a hierarchical clustering of users and rated items. Both users and items are clustered independently into two cluster hierarchies. Each user and each item belongs to exactly one leaf of a cluster hierarchy. Each node of a cluster hierarchy contains the members of descending nodes. The roots of the hierarchy contain all users and items respectively. The hierarchies are derived, so that clusters contain similar items or users, and the degree of similarity increases as we go down the hierarchy from the root to the leaves.

Row vectors of the rating matrix $score(user, item)$ (expressing all ratings to all items by a given user) are clustered in the user-cluster-hierarchy and column vectors (expressing all ratings to a given item by all users) are clustered in the item-cluster-hierarchy.

2.1 Deriving Cluster Hierarchies

The goal of clustering vectors is to group similar data points. In our case the data point are the rating vectors of users (or in the case of a rated-item cluster the rating vectors of the item). Similarity between data points (users or items) is determined by a distance function (which is described in the next section). The center of a cluster is determined by the average of all contained data points. The quality of a clustering is expressed as distortion, which is the sum of the distances of all data-points from the center of their assigned cluster.

Cluster hierarchies can be constructed by using either a top-down or a bottom-up approach. The bottom-up approach is not feasable in our case due to the size of the database. Therefore, we decided to construct the hierarchy of clusters using the top-down approach. The steps of our top-down algorithm are listed below:

- Initially each item is assigned to a single cluster, which is the root of the hierarchy R_C.
- For R_C the distortion is determined. If the distortion per item is greater than a predefined threshold this cluster is further split into two son clusters C_1 and C_2.
  - From a random sample set of items in R_C the pair I_1 and I_2 with the maximum distance is determined. I_i are assigned to C_i respectively. All the remaining items are assigned to the closest cluster of C_i.
  - While items in one son-cluster are closer to the other son-cluster the items are moved respectively.
- For each son-cluster all the steps above are performed as if they were roots.

2.2 Clustering Metrics

We use the mean-squared-difference as the distance between items and clusters (see equation 3). However, rating vectors are incomplete, since users have not rated all items and items have not been rated by everyone. Default values have to be assumed for the gaps in the vectors associated with items. In contrary of assuming an arbitrary value for all missing components, our algorithm chooses the center of a common parent cluster as a default for undefined components in vectors. If the component of the center of a cluster is not defined, then all items within this cluster have not defined this component (otherwise the center would be defined for this dimension).

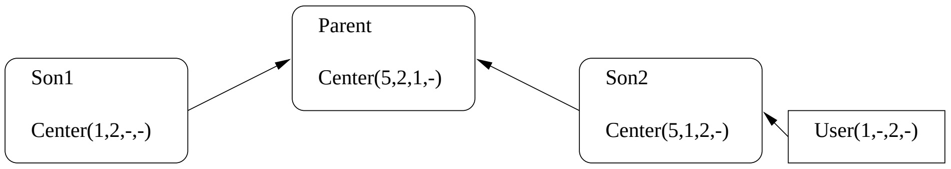


Figure 1: Default value inheritance

Figure 1 depicts an example clustering. If in this example it should be decided whether $user$ has to be moved to $son1$ both distances $d(user, son1)$ and $d(user, son2)$ need to be compared. For the completion of their vectors $user$ inherits 2 as a default for the second component from $parent$ and $son1$ inherits 1 as the third component of its center. After assuming the defaults the first three dimensions of $son1$, $son2$ and $user$ are defined and the distance can be determined.

The inheritance of defaults takes advantage of the fact, that the neighborhood within a cluster hierarchy contains similar items, and is therefore more precise than assuming a global default. Consequently less noise is introduced to the calculation of distances.

2.3 Cluster Prediction

Cluster hierarchies of desired fine granularity can be calculated by using the previously described clustering algorithm. In this section we describe how the cluster hierarchies can be used to obtain predictions. If a rating $p(u, f)$ for a user u and an item f should be provided, the following formulas are used:

$$\begin{aligned}
C_i &: \text{component } i \text{ of center vector of } C && (1)\\
I_C &= \{i | C_i \text{ is defined}\} && (2)\\
d(C,u) &= \frac{\sum_{i \in I_C \cup I_u} (u_i - C_i)^2}{|I_C \cup I_U|} \text{ (using default value inheritence for undefined ratings)} && (3)\\
H_U &: \text{cluster hierarchies for users } (H_F \text{ for items}) && (4)\\
L_u &: \text{leaf cluster of } H_u \text{ of user } u && (5)\\
P_u &: \text{path of all clusters from the root of } H_U \text{ to the leaf } L_u \text{ (same for } P_f) && (6)\\
P_{u/f} &= \{C \in P_u | C_f \text{ is defined.}\} \text{ similarly for } P_{f/u} && (7)\\
DI_C &= \sum_{i \in C} \frac{d(C,i)}{|C|} && (8)\\
DI_{H_U} &= DI(\text{root of } H_U) && (9)\\
w_U(C) &= 1 - \frac{DI_C}{DI_{H_U}} && (10)\\
p(u,f) &= \frac{\sum_{C \in P_{u/f}} (C_f * w_U(C)) + \sum_{c \in P_{f/u}} (C_u * w_F(C))}{\sum_{C \in P_{u/f}} w_U(C) + \sum_{C \in P_{f/u}} w_F(C)} && (11)
\end{aligned}$$

The last formula performs the actual prediction. Expressed in words, the prediction formula is the weighed sum of the defined centers of all nodes in the cluster hierarchies on the paths from the hierarchy roots to the particular leaves. The weights are derived from the distortions of the hierarchy nodes.

3 Modeling Problematic Cases

For evaluating different CF-algorithms, we performed simulations on a dataset of movie ratings. We used the data collected by Digital Equipment Research Center from 1995 to 1997 in a collaborative movie recommendation project. The EachMovie dataset contains about 2.8 million ratings of 60000 users for 1600 movies on a rating scale from 0 to 5. After preprocessing, the dataset contains 2.5 million ratings. For each user, ratings were split into 80% training-set and 20% test-set. From the ratings in the training-set subsets are derived thus certain model parameters are met. Then the suitable subset of the test-set is predicted, using the described algorithms. The performance of an algorithm is measured in terms of the mean-absolute-error(MAE=$\frac{\sum |p_i - r_i|}{n}$) between the predictions and the ratings in the test-set. Even though this measure strongly depends on the rating scale in use and can therefore not be used to compare single results with results of other studies, we found this measure indicative enough to compare the algorithms within this paper.

3.1 The *Bootstrap-Case*

When a CF-application is set up, it has initially no users but many items, which can be rated, or for which recommendations should be predicted. Eventually, some users will start to add ratings. We captured the number of initial users (IU) as a parameter for characterizing a CF-database. Users, who start using a CF-application, are confronted with many unrated items, of which they rate eventually a portion. The ratio of rated items to the number of items increases over the time. The amount of ratings a user has provided that we call Rating Frequency (RF) is used to express that behavior. It is important for service providers, who want to start a new CF-service, to understand the impact of IU

and *RF* on the quality of the prediction, so that they can evaluate how much incentive they should give to initial users. These incentives are part of the cost required to set up an efficient CF-service. This is the reason why in the *Bootstrap-Case*, we compare prediction algorithms for low values of *IU* and *RF*.

3.2 Simulation of the *Bootstrap-Case*

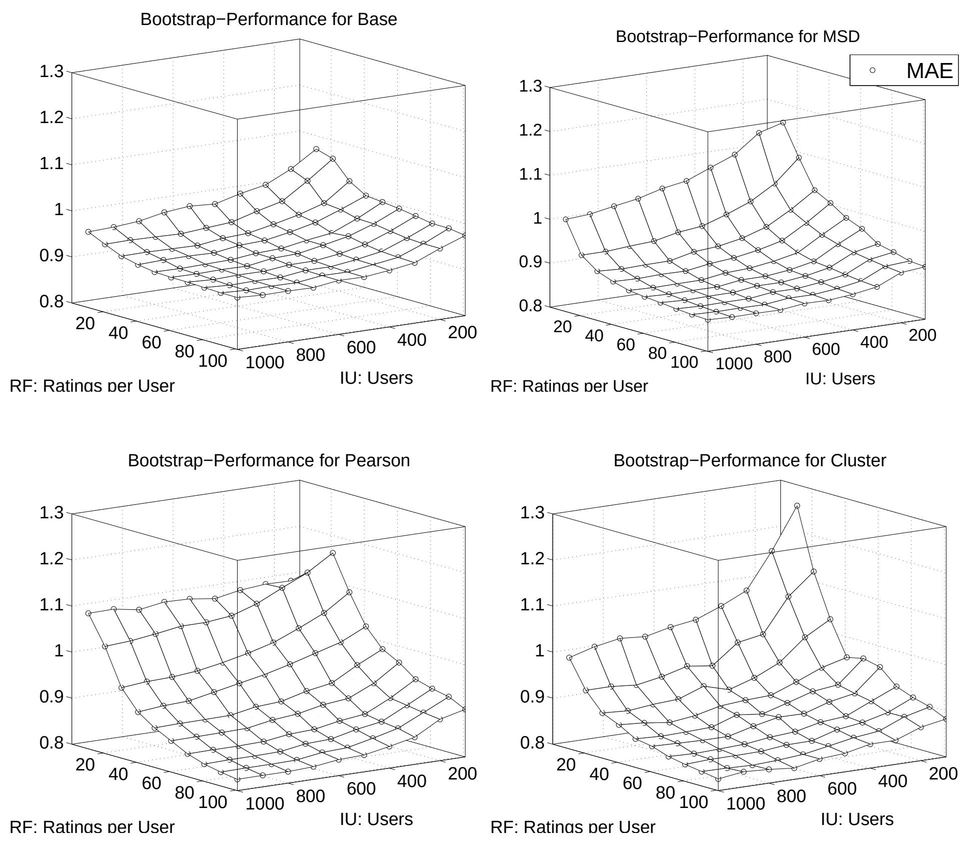


Figure 2: The CF-algorithms in the frame of the *Bootstrap-Case*: Precision.

Figure 2 depicts measurements of the mean absolute prediction error (MAE) for varying parameters of the *Bootstrap-Case* for all algorithms under study. In this experiment the database is modeled so that it satisfied the varying settings of the parameters for *RF* and *IU*. It is notable that the Cluster and Pearson algorithm perform equally for high values of *RF* and *IU*. However, the Cluster algorithm converges faster to the maximum performance and is therefore more suitable for sparse CF-databases. These graphs also indicate that, for low *RF* and *IU* (small CF-databases), the CF-algorithms under study generally perform poorly. In such cases the prediction precision becomes very low. In these examples, the Pearson algorithm, which is commonly used for collaborative filtering [3, 4, 5], performs worse than the Base algorithm. These graphs still do not enable a simple comparison of different algorithms for collaborative filtering. The next section performs a cost analysis based on the results of the *Bootstrap-Case* simulation, which is of more practical interest to determine the best algorithm to use for providers of CF-services.

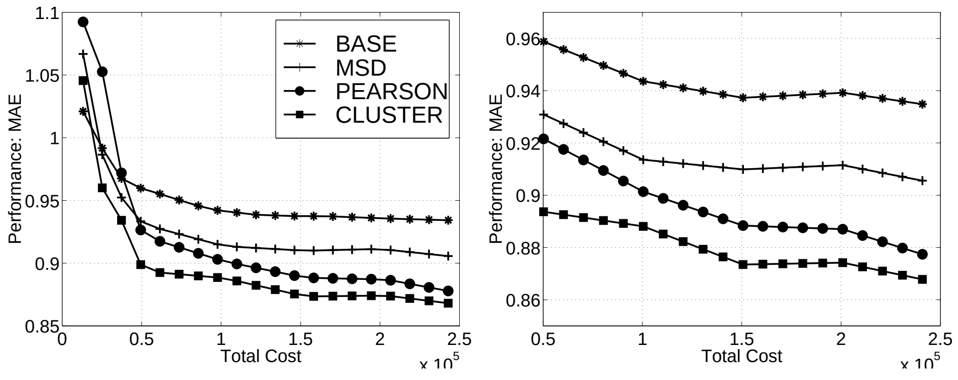
3.3 Cost analysis of the *Bootstrap-Case*

A provider of a CF-service needs to initialize(bootstrap) the CF-database, so that his service can be attractive to users. A low amount of data in the CF-database leads to a poor prediction precision and therefore to an unattractive service. The service provider needs to know how much he has to invest to bootstrap his system in order to provide a desired prediction quality. As has been shown, CF-databases can be grown in two dimensions, *RF* and *IU*. Each direction might involve different costs, e.g. for paying people to rate items. We identify the cost for one rating as C_R, since the effort to provide ratings with no predictions in return has to be compensated. Also a rater has to be attracted to provide ratings, e.g. by advertisement. We identify the cost to attract a new rater as C_U. The total cost of a bootstrap can now be expressed as in the following formula:

$$totalcost(IU, RF, C_U, C_R) = IU \cdot RF \cdot C_R + IU \cdot C_U$$

Depending on the settings of the parameters C_U and C_R, the total cost of bootstrapping a CF-database to a sufficiently performing level can vary. As we have shown previously different CF-algorithms reach differing levels of performance with the same given CF-databases. Generally two characteristic configurations of the cost parameters can be identified:

- $C_U = 0$ and $C_R > 0$: Raters can be attracted easily, for example they are users connecting from the Internet after having seen an advertisement, but they need to be compensated for each rating.
- $C_U \gg C_R > 0$: Raters are hard to obtain, for example they are hired by the provider, but the cost of compensation for each rating is considerably smaller.



(a) $C_U = 0, C_R = 5$ (b) $C_U = 200, C_R = 1$

Figure 3: Precision as function of Total-Cost of bootstrapping a CF-Database.

Figures 3 illustrate the performance of the algorithms under study as functions of the total cost for bootstrapping a CF-database. In both plots the Cluster-algorithm performs better than the others. That means CF-databases can be bootstrapped more inexpensively if the Cluster algorithm is used. Also it is notable that the curve of the Cluster-algorithm in the second pricing model has initially a smaller slope than the other algorithms. This fact indicates that the Cluster-algorithm favors the increase of RF rather than IU.

3.4 The *New-User-Case*

New users in a CF-application are characterized by a low amount of rated items, which corresponds to their RF. Even though the CF-database might be sufficiently filled, the prediction performance for new users is expected to be rather poor, since most algorithms rely on the fact that similar users can be detected in the database. If a new user increased his RF by rating more items, then the prediction precision, computed based on his ratings, can be expected to increase.

For investigating the *New-User-Case*, we used a fully-grown CF-database. For some users (new users) ratings were removed from the database. Later the removed ratings were incrementally added to the CF-database (new users who rate more and more items). In each step the prediction precision is measured for different algorithms for the new users.

The graphs of figure 4 plot the MAE as an indicator for precision for new users as functions of their RF for different CF-algorithms. Generally, for low amounts of initial votes the prediction precision is poor for the algorithms under study. As the graphs of figure 4 indicate, higher precision can be obtained by using the Cluster algorithm. Especially, for lower RF the precision difference between the

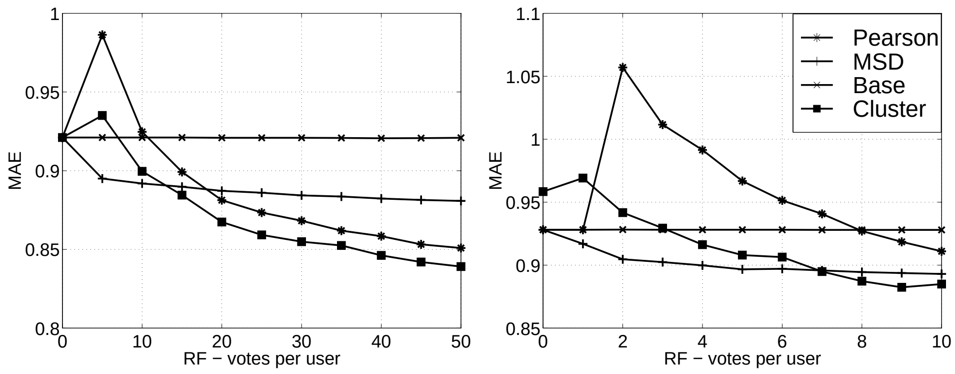


Figure 4: Prediction precision of CF-algorithms in the frame of the *New-User-Case*.

Pearson and Cluster algorithms notable high. The difference diminishes with increasing RF. This result indicates that for users with few ratings the Cluster algorithm should be used to furnish predictions. It is also notable that the Cluster and Pearson algorithms become chaotic for very low RF. Then the MSD or the Base algorithm provide better results.

4 Conclusion

This paper makes three contributions:

First, it presents a new approach to collaborative filtering, the Cluster algorithm, which is designed to perform better in case of sparse CF-databases. A heuristic is presented, which is capable of clustering large dimensional sparse vectors as rating vectors. Further a prediction algorithm is presented.

Second, it describes two cases of sparse CF-databases, which are of importance, the *Bootstrap-Case* and the *New-User-Case*. It identifies parameters, which allow to model these problematic cases for simulations. Further it presents a cost analysis, which allows to select the right algorithm for a given cost and performance requirement.

Third, it experimentally validates the proposed Cluster algorithm by comparing its performance with classically used algorithms for the described sparsity cases of CF-databases.

Acknowledgments: This research was supported by a regional research grant, and by Eurecom's industrial members: Ascom, Cegetel, France Telecom, Hitachi, IBM France, Motorola, Swisscom, Texas Instruments, and Thomson CSF. The dataset used for this paper was generously provided by Digital Equipment Corporation (EachMovie (See `http://www.digital.research.com/SRC/EachMovie/`.).

References

[1] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, Madison, WI, July 1998. Morgan Kaufmann Publisher.

[2] J. Delgado, N. Ishii, and T. Ura. Content-based collaborative information filtering. In *Cooperative Information Agents II*, Lecture Notes in AI 1435, pages 206–215. Springer Verlag, 1998.

[3] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordan, and J. Riedl. Grouplens: Applying collaborative filtering to usenet news. *CACM*, 40(3), March 1997.

[4] B.M. Sarwar, J. A. Konstan, A. Bochers, J. Herlocker, B. Miller, and J. Riedl. Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *Proceedings of ACM CSCW'98*, 1998.

[5] U. Shardanand. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of Human Factors in Computing Systems, CHI '95*, 1995.