

USER PROFILING: COLLABORATIVE FILTERING

Miha Grčar

Department of Knowledge Technologies
Jozef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel: +386 31 657881; fax: +386 1 4251038
e-mail: miha.grcar@ijs.si

ABSTRACT

Collaborative filtering is based on the assumption that “similar users have similar preferences”. In other words, by finding users that are similar to the active user and by examining their preferences, the recommender system can (i) predict the active user’s preferences for certain items and (ii) provide a ranked list of items which active user will most probably like. Collaborative filtering generally ignores the form and the content of the items and can therefore also be applied to non-textual items. Furthermore, collaborative filtering can detect relationships between items that have no content similarities but are linked implicitly through the groups of users accessing them. These groups (communities) are formed around a specific user profile.

1 INTRODUCTION

Collaborative filtering compares users according to their preferences. Therefore, a database of users’ preferences must be available. The preferences can be collected either explicitly (explicit rating) or implicitly (implicit rating). In the first case the user’s participation is required. The user explicitly submits his/her rating of the given item. Such rating can, for example, be given as a score on a rating scale from 1 to 5. The implicit ratings, on the other hand, are derived from monitoring the user’s behavior. In the context of the Web, access logs can be examined to determine such implicit preferences. For example, if the user accessed the document, he/she implicitly rated it 1. Otherwise the document is assumed to be rated 0 by the user (i.e. “did not visit”).

The collaborative filtering process can be divided into two phases: (i) the model generation phase and (ii) the recommendation phase. Algorithms which tend to skip the first phase are the so called memory-based approaches (also referred to as lazy learning approaches or the nearest neighbors algorithms) (see Section 2). The preferences database is a huge user-by-item matrix, $R = [r_{i,j}]$, constructed from the data at hand. A matrix element $r_{i,j}$ represents user i ’s rating of item j . Memory-based approaches search the matrix for relationships between users and/or items. Model-based approaches, on the other hand, use the data from R to build a model that enables faster and more accurate

recommendations (see Sections 3–6). The model generation is usually performed offline over several hours or days.

When dealing with collaborative filtering, two fundamental problems of collaborative filtering have to be taken into account: (i) the sparsity of the data and (ii) the scalability problem. The first problem, which we encounter when R is missing many values, can be partially solved by incorporating other data sources (such as the contents of the items) [2], by clustering users and/or items [3, 4], or by reducing the dimensionality of the initial matrix (see Section 3). The last two techniques also counter the scalability problem. This problem arises from the fact that the basic nearest neighbor algorithm fails to scale up its computation with the growth of the number of users and the number of items. Some of the approaches for countering the two problems are described in Section 3.

2 MEMORY-BASED APPROACH TO COLLABORATIVE FILTERING

A straightforward algorithmic approach to collaborative filtering involves finding k nearest neighbors (i.e. the most similar users) of the active user and averaging their ratings of the item in question. Even better, we can calculate weighted average of the ratings, weights being similarity, correlation or distance factors (later on in the text the term *similarity* is used to denote any of the three measures) between a neighbor-user and the active user [e.g. 3]. We can look at a user as being a feature vector. In this aspect, items that are being rated are features and ratings given by the user to these items are feature values. The following formula can be applied to predict user u ’s rating of item i :

$$p_{u,i} = \bar{v}_u + \kappa \sum_{j \in \text{Users}} w(u, j) (v_{j,i} - \bar{v}_j) \quad (1)$$

where $w(u_1, u_2)$ is the weight which is higher for more similar, less distant or more correlated users (feature vectors), \bar{v}_u is the mean rating given by user u , $v_{j,i}$ is the rating of item i given by user j , and κ is merely a normalization factor which depends on our choice of weighting.

When representing a user as a feature vector, many of the features have missing values, since not every item was explicitly rated by the user. This fact introduces the sparsity

problem which implies that measuring similarity between two feature vectors is not a trivial task. Many times two feature vectors have only a few or no overlapping values at all. When computing similarity over only a few values, the similarity measure is unreliable. Furthermore, when there is no overlapping between two vectors, the degree of similarity can not be determined.

The equation (1) was introduced by [5]. If no ratings of item i are available, the prediction is equal to the average rating given by user u . This is an evident improvement of the equation that simply calculates weighted average.

2.1 Weight Computation

The weights can be defined in many different ways. Some of the possibilities are summarized in the following sections.

2.1.1 Cosine Similarity

The similarity measure can be based on the cosine of the angle between two feature vectors. This technique was primarily used in information retrieval for calculating similarity between two documents, where documents were usually represented as vectors of word frequencies. In this context, weights can be defined as:

$$w(u_1, u_2) = \frac{\sum_{i \in \text{Items}} \frac{v_{u_1,i} v_{u_2,i}}{\sqrt{\sum_{k \in I_1} v_{u_1,k}^2} \sqrt{\sum_{k \in I_2} v_{u_2,k}^2}}$$

2.1.2 Pearson Correlation

Weights can be defined in terms of the Pearson correlation coefficient [5]. Pearson correlation is also used in statistics to evaluate the degree of linear relationship between two variables. It ranges from -1 (a perfect negative relationship) to $+1$ (a perfect positive relationship), with 0 stating that there is no relationship whatsoever. The formula is as follows:

$$w(u_1, u_2) = \frac{\sum_{j \in \text{Items}} (v_{u_1,j} - \bar{v}_{u_1})(v_{u_2,j} - \bar{v}_{u_2})}{\sqrt{\sum_{j \in \text{Items}} (v_{u_1,j} - \bar{v}_{u_1})^2} \sqrt{\sum_{j \in \text{Items}} (v_{u_2,j} - \bar{v}_{u_2})^2}}$$

2.1.3 Weight Amplification and Inverse User Frequency

We can have good confidence in the computed weight in the case when a lot of overlapping values are available. On the other hand, if there are only few overlapping values, the weight's reliability is questionable. To incorporate the degree of confidence into the equation (1), we can lower the weights that are based on only few items and vice versa [e.g. 4]. Additionally, we can amplify weights [3]. This means that we reward weights that are close to 1 and punish those that are close to 0. Another approach for bettering the weights is also the application of the inverse user frequency as described in [3]. The main idea is that universally liked items are less relevant for predictions than those that are popular with a smaller number of people. Therefore, we transform each rating by multiplying it with the inverse user frequency which is defined as $\log n/n_j$, where n_j is the

number of users who have rated item j and n is the total number of users.

2.1.4 Default Rating

The problem with the Pearson correlation formula is that only the overlapping ratings can be used for computation. Due to high sparsity of the data, the number of overlapping ratings is rather small. If user A is overlapping with user B in items 1, 2 and 7, and user B is overlapping with user C in items 4, 8 and 12, but users A and C are not overlapping in their rated items, then no relationship can be detected between users A and C. In other words, using Pearson correlation we cannot detect transitive relationships. To avoid this problem, we introduce a slightly modified equation, referred to as default rating. Instead of considering the intersection of available ratings from both users, we now take their union and fill in the missing values with some predefined default value d [3]. At this point we could also consider filling in the user's average rating instead of the constant d . The missing ratings could also be predicted by averaging the ratings of the items that have similar content. The latter possibility is explored by the so called content-boosted collaborative filtering [2].

3 DIMENSIONALITY REDUCTION TECHNIQUES

We are initially dealing with a huge user-by-item matrix. Since there can be millions of users and millions of items, the need to reduce the dimensionality of the matrix emerges. The reduction can be carried out by selecting only relevant users (instance selection) and/or by selecting only relevant items (feature selection). Other forms of dimensionality reduction can also be employed, as described later on in this section.

It is shown by some researchers that feature selection, instance selection and other dimensionality reduction techniques not only counter the scalability problem but also result in more accurate recommendations [4, 6, 8]. Furthermore, the sparsity of the data is consequentially decreased.

When reducing the dimensionality, the first possibility that comes to mind is the removal of the users that did not rate enough items to participate in collaborative filtering. From the remaining users, we can randomly choose n users to limit our search for the neighborhood of the active user. This method is usually referred to as random sampling. Also, rarely rated items can be removed for better performance. Still, these relatively simple approaches are usually not sufficient for achieving high scalability and maintaining the recommendation accuracy.

3.1 Latent Semantic Analysis (LSA)

A more sophisticated dimensionality reduction approach is called Latent Semantic Analysis (LSA) [9]. It is based on Singular Value Decomposition (SVD) of the user-by-item matrix. By using linear algebra, a matrix can be decomposed into a triplet, namely $M = U\Sigma V^T$. The diagonal matrix Σ holds the singular values of M . If we set all but K largest singular values to zero and thus obtain Σ' , we can approximate M as $M' = U\Sigma'V^T$. By doing so we transform our initial high-dimensional matrix into a K -dimensional (low-dimensional) space. The neighborhood of the user can now be determined by transforming the user vector into the low-dimensional space of the approximated matrix and finding k nearest points representing other users. Searching through a low-dimensional space is clearly faster. Furthermore, dimensionality reduction reduces sparsity and captures transitive relationships among users. This results in higher accuracy.

3.2 Probabilistic Latent Semantic Analysis (pLSA)

On the basis of LSA, Probabilistic Latent Semantic Analysis (pLSA) was presented [7]. pLSA has its roots in information retrieval but can also be employed for collaborative filtering [6]. In a statistical model, an event like “person u ‘clicked on’ item i ” is presented as an observation pair (u, i) (note that in such case we are dealing with implicit ratings). User u and item i “occur” paired with a certain probability: $P(u, i)$. We are in fact interested in the conditional probability of item i occurring given user u : $P(i | u)$. This conditional form is more suitable for collaborative filtering since we are interested in the active user’s interests.

The main idea of an aspect model (such as pLSA) is to introduce a latent variable z , with a state for every possible occurrence of (u, i) . User and item are rendered independent, conditioned on z : $P(u, i) = P(z)P(u | z)P(i | z)$. $P(i | u)$ can be written in the following form:

$$P(i | u) = \sum_z P(i | z)P(z | u) \quad (2)$$

Note that we limit the number of different states of z so that it is much smaller than the number of (u, i) pairs. Let us denote the number of users with N_u , the number of items with N_i , and the number of different states of z with N_z , where $N_z \ll N_u, N_i$. We can describe the probabilities $P(i | u)$ with $S_1 = N_i \times N_u$ independent parameters. On the other hand, we can summarize the probabilities $P(i | z)$ and $P(z | u)$ with $S_2 = N_i \times N_z + N_u \times N_z$ independent parameters. The dimensionality reduction is evident from the fact that $S_2 < S_1$ (if N_z is small enough). Such latent class models tend to combine items into groups of similar items, and users into groups of similar users. In contrast to clustering techniques (see Section 6), pLSA allows partial memberships in clusters (clusters being different states of z).

In equation (2), the probabilities $P(z | u)$ and $P(i | z)$ can be determined by the Expectation Minimization algorithm using various mixture models. To support explicit ratings, we extend pLSA by incorporating rating to our observation

pair and thus observing triplets of the form (u, i, r) , where r represents a rating score.

The relation of this method to LSA and SVD can be explained by representing the probabilities $P(u, i)$ in the form of a matrix M_p which can be decomposed into three matrices, namely $M_p = U_p \Sigma_p V_p^T$. The elements of these matrices are $u_{i,k} = P(u_i | z_k)$, $\sigma_{k,k} = P(z_k)$, $v_{j,k} = P(i_j | z_k)$ [1].

4 COLLABORATIVE FILTERING AS A CLASSIFICATION TASK

The collaborative filtering task can also be interpreted as a classification task, classes being different rating scores [11]. Virtually any supervised learning algorithm can be applied to perform classification (i.e. prediction). For each user we train a separate classifier. A train set consists of feature vectors representing items the user already rated, classifications being rating scores from the user. Clearly the problem occurs if our training algorithm cannot handle missing values in the sparse feature vectors. It is suggested by [11] to represent each user by several instances (optimally, one instance for each possible rating score). On a 1–5 rating scale, user A would be represented with 5 instances, namely A -rates-1, A -rates-2, ..., A -rates-5. The instance A -rates-3, for example, would hold ones (‘1’) for each item that user A rated 3 and zeros (‘0’) for all other items. This way, we fill in the missing values. We can now use such binary feature vectors for training. To predict a rating, we need to classify the item into one of the classes representing rating scores. If we wanted to predict scores on a continuous scale, we would have to use a regression approach instead of classification.

5 ITEM-BASED COLLABORATIVE FILTERING

All the collaborative filtering approaches we have discussed so far are user-centric in the way that they concentrate on determining the user’s neighborhood. Some researchers also considered item-based collaborative filtering [12]. The main idea is to compute item-item similarities (according to the users’ ratings) offline and make use of them in the online phase. To predict user u ’s rating of item i , the online algorithm computes a weighted sum of the user u ’s ratings over k items that are most similar to item i . The main question in this approach is how to evaluate item-item similarities to compute a weighted sum of the ratings. Item-item similarities can be computed using the techniques for computing user-user similarities, described in Section 2.1. The winning technique, according to [12], is the so called adjusted cosine similarity measure. This is a variant of cosine similarity which incorporates the fact that different users may have different rating scales. The similarity measures are then used as weights for calculating a weighted sum of k nearest items.

6 SOME OTHER APPROACHES

Let us briefly summarize some other techniques. Interested reader should consider the appropriate additional reading.

6.1 Horting

Horting is a graph-theoretic approach to collaborative filtering [13]. It involves building a directed graph in which vertices represent users and edges denote the degree of similarity between them. If we are trying to predict user u 's rating of item i , we need to find a directed path from user u to a user who has rated item i . By using linear transformations assigned to edges along the path, we can predict user u 's rating of item i . No other user along this path rated item i . This means that horting also explores transitive relationships between users.

6.2 Clustering Techniques

Bayesian and non-Bayesian clustering techniques can be used to build clusters (or neighborhoods) of similar users [3, 4, 6]. The active user is a member of a certain cluster. To predict his/her rating of item i , we compute the average rating for item i within the cluster that the user belongs to. Some such methods allow partial membership of the user in more than one cluster. In such case, the predicted rating is summed over several clusters, weighted by the user's participation degree. Clustering techniques can also be used as instance selection techniques (instances being users) that are used to shrink the candidate set for the k nearest neighbors algorithm.

6.3 Bayesian Networks

Bayesian networks with a decision tree at each node have also been applied to collaborative filtering [3, 14]. Nodes correspond to items, and states of each node correspond to possible rating scores. Conditional probabilities at each node are represented in a form of decision trees in which nodes again are items, edges represent preferences, and leaves represent possible states (i.e. rating scores). Bayesian networks are built offline over several hours or even days. This approach is not suitable in systems that need to update rapidly and frequently.

7. ACKNOWLEDGEMENTS

I would like to thank Marko Grobelnik and Dunja Mladenić for their mentorship and to Tanja Brajnik for every minute she invests in my English.

References

[1] P. Baldi, P. Frasconi, P. Smyth. Modelling the Internet and the Web. pp. 171–209. ISBN: 0-470-84906-1. 2003.
[2] P. Melville, R. J. Mooney, R. Nagarajan. Content-Boosted Collaborative Filtering for Improved Recommendations. *Proceedings of the Eighteenth*

National Conference on Artificial Intelligence. pp. 187–192. 2002.
[3] J. S. Breese, D. Heckerman, C. Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. 1998.
[4] C. Zeng, C.-X. Xing, L.-Z. Zhou. Similarity Measure and Instance Selection for Collaborative Filtering. *Proceedings of the Twelfth International World Wide Web Conference*. 2003.
[5] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering for Netnews. *Proceedings of CSCW '94*. 1994.
[6] T. Hoffman. Latent Semantic Models for Collaborative Filtering. *ACM Transactions on Information Systems*. Vol. 22. Issue 1. pp. 89–115. 2004.
[7] T. Hoffman. Probabilistic Latent Semantic Analysis. *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*. 1999.
[8] K. Yu, X. Xu, M. Ester, H.-P. Kriegel. Selecting Relevant Instances for Efficient and Accurate Collaborative Filtering. *Proceedings of the Tenth International Conference on Information and Knowledge Management*. Session: Collaborative Filtering and Algorithms. pp. 239–246. 2001.
[9] S. Deerwester, S. T. Dumais, R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the Society for Information Science*. Vol. 41. Issue 6. pp. 391–407. 1990.
[10] K. Nigam, A. K. McCallum, S. Thrun, T. Mitchell. Text Classification from Labeled and Unlabeled Documents Using EM. *Machine Learning*. Vol. 39. Issue 2–3. pp. 103–134. 2000.
[11] D. Billsus, M. J. Pazzani. Learning Collaborative Information Filers. *Proceedings of the Fifteenth International Conference on Machine Learning*. 1998.
[12] B. Sarwar, G. Karyps, J. Konstan, J. Riedl. Item-Based Collaborative Filtering Recommendation Algorithms. *Proceedings of the Tenth International Conference on World Wide Web*. pp. 285–295. 2001.
[13] C. C. Aggarwal, J. L. Wolf, K.-L. Wu, P. S. Yu. Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering. *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pp. 201–212. 1999.
[14] D. M. Chickering, D. Heckerman, C. Meek. A Bayesian Approach to Learning Bayesian Networks with Local Structure. *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*. 1997.