Learning What People (Don't) Want

Thomas Hofmann

Department of Computer Science Brown University
th@cs.brown.edu

Abstract. Recommender systems make use of a database of user ratings to generate personalized recommendations and help people to find relevant products, items, or documents. In this paper, we present a probabilistic, model-based framework for user ratings based on a novel collaborative filtering technique that performs an automatic decomposition of user preferences. Our approach has several benefits, including highly accurate predictions, task-optimized model learning, mining of interest groups and patterns, as well as a highly efficient and scalable computation of predictions and recommendation lists.

1 Introduction

Knowing what one really wants is far from obvious. In order to figure out which book to read, which movie to watch, or which Web site to visit, we often rely on advise given to us by other people. Yet, in many situations one would like to automate the process of recommending items by anticipating user preferences. For example, in an electronic commerce setting customer may want to automatically receive accurate recommendations. Similarly, in the case of news groups or Web communities one would like to have some form of automated support, since these user groups are often far too large to allow its members to directly share experiences or to interchange recommendation. And even if one is lucky enough to get a person's advise on a subject, the question of who's advise to trust on which issue always remains a crucial question.

Recommender systems have been proposed and utilized in this context to leverage existing user data (ratings, profiles, logs) and to help people share their evaluations. Prominent Internet sites like amazon.com or CDnow use such systems for personalized product recommendation and to complement the traditional content-oriented search functionality. The predominant technique that powers recommender systems is *collaborative filtering* [2] and the most popular methods are variations around the memory-based approach proposed in the GroupLens [10,8] project. The underlying principle is extremely simple - for a given (so-called *active*) user, find people with similar interests and use their ratings and judgements to recommend new items.

In this paper, we propose a radically different approach to collaborative filtering which uses a sparse matrix decomposition technique called Probabilistic Latent Semantic Analysis [5] to automatically discover preference patterns in user profile data. In order to motivate our approach, we will first briefly discuss the dominant paradigm for collaborative filtering in Section 2 and point out what we think are its conceptional flaws. Section 3 presents our approach whereas Section 4 deals with an experimental evaluation and discussion.

L. De Raedt and P. Flach (Eds.): ECML 2001, LNAI 2167, pp. 214–225, 2001.

2 Memory-Based Approaches

The by far most popular class of algorithms for collaborative filtering relies on the following principle. Given an active user a and a database of user profiles, recommendations are determined by, (i) computing the similarity between the active user and all the users in the database, (ii) for each item, forming a weighted and properly normalized vote over all (or some) users in the database, where the weights reflect the similarity computed in (i).

More specifically, the Pearson correlation has been proposed in many landmark papers [8,11], which computes the similarity between user profiles as

$$w(a,u) = \sum_{y \in V_a \cap V_u} \frac{(v_{a,y} - \bar{v}_a)(v_{u,y} - \bar{v}_u)}{\sqrt{\sigma_a \sigma_u}}. \tag{1}$$

Here $v_{u,y}$ denotes the (known) vote of user u on item y, while $\bar{v}_u$ and σ_u refer to the mean and standard deviation of votes of user u, respectively. Based on these weights a predictive vote for the active user is computed as follows,

$$p_{a,y} = \frac{\sum_{u:y \in V_u} w(a,u) v_{u,y}}{\sum_{u:y \in V_u} |w(a,u)|}. \tag{2}$$

Variations of this basic methodology include the use of different correlation measures like the Spearman correlation [4], the thresholding of weights, and the inhomogeneous weighting of items. These methods are often called *memory-based* methods or neighbor-based methods, since predicted votes are computed directly from the database of user profiles.

What is the fundamental assumption underlying this class of algorithms? Most importantly in our opinion, it is the idea of a *constant similarity* measure between user profiles. Here "constant" refers to the fact that $w(a,u)$ does not depend on the actual item y under consideration.[1] This is a very strong assumption, since it presupposes that in making a prediction for an active user, the "degree of trust" in the rating of some other user does not depend on the specific item. Similar assumptions are made be many model-based and hybrid approaches, e.g., the clustering approach of [12] and the personality diagnostics approach in [9]. The dependency network approach proposed in [3] on the other hand, builds a predictive model for each item and hence does not suffer from this limitation.

Potential problems that come with the above assumption can best be illustrated with an example: imagine we would like to recommend music to an active user A who likes opera. In the database, we find a very "post-modern character", B, who also adores opera but at the same time likes Salsa rhythms. Should one recommend Salsa to the purist opera fan? Or should one effectively eliminate B from the neighbor set of A, although B's opera recommendations could be highly relevant? Similarly, in the opposite situation, imagine that "typically" opera and Salsa are negatively correlated and most people who like one genre

[1] Only the overall normalization in Eq. (2) is different for every item, although one can hardly consider this to be an "item-specific" weighting.

dislike the other one. This would imply that recommending items of one or the other type to B, but never both, dependent on the exact similarity weights and proportions.

Intuitively, what is missing in standard memory-based approaches is the possibility to model that fact that one person is a reliable recommender for another person with respect to a *subset of items*, but not necessarily for all possible items. This deficit is expected to become more relevant with increasing diversity of the item set and users' interest patterns. In general, we strongly believe that the simple notion of similarity between one person and another person fails to capture the multi-dimensional nature of human preferences.

We see the solution to this problem in a method that couples recommendation with a decomposition of preferences, where each preference pattern models an interest or trait shared among a community of people. There is a reciprocal characterization of user communities by sets of items and vice versa, along with the typical ratings that describe the nature of the relation, be it positive, neutral, or negative.

3 Decomposing Preferences

We propose a probabilistic model-based approach to collaborative filtering which overcomes the major limitations of memory-based methods and offers a number of additional advantages:

- User profiles are explicitly decomposed into statistically significant *interest patterns*, each pattern typically dealing with a subset of the items. This allows to selectively share recommendations among users by introducing a dependency on the specific item or type of item, thereby increasing accuracy.
- The preference decomposition also perform *data mining* by revealing hidden patterns that drive user ratings. This can be used for identifying user communities as well as for data visualization.
- The probabilistic preference model can be used to tailor recommender systems to the task at hand. Often items come with specific costs or utilities and one can optimize the recommendations to maximize the expected utility.
- Once the model is trained, the database is no longer required to make recommendations. A model-based approach thus avoids the need to have a potentially huge database available to make recommendations. This has advantages in terms of memory requirements as well as system speed.

3.1 Latent Class Models for User Ratings

There are two variants of probabilistic models that we investigate in this paper. They correspond to different settings of the recommendation problem: (I) predicting a user rating for an unknown item, i.e., estimating $P(v|u, y)$ or $p_{u,y} \equiv \sum_v v\, P(v|u, y)$ and (II) predicting a rating in conjunction with a selection, i.e., estimating $P(v, y|u)$. Which setting is more appropriate depends on the actual application and objective. For example, in an electronic commerce

setting, $P(y|u)$ may model the fact whether a user is likely to purchase a product, which may be an important part of the system, irrespective of the actual user satisfaction expressed by the rating v.

Formally, let us introduce a latent variable $z_{u,y}$ for each (actual and potential) observation triplet $(u, y, v_{u,y})$, thus defining the complete data of an elementary rating event by $(u, y, v_{u,y}, z_{u,y})$. Intuitively, each quadruple is supposed to model the fact that a person u votes on item y with rating $v_{u,y}$, "because of" $z_{u,y}$. Each latent variable is intended to offer a hypothetical explanation for a rating. For example, a high rating of a person on a piece of music, say "Fidelio", might be explained by the fact that the person is interest in the particular genre (opera). However, an alternative explanation could be that a person likes works by a specific composer (Beethoven) and the genre might be completely irrelevant. Yet a third person might like "Fidelio" because of its political message and its praise of freedom and love. Of course, it is not clear *a priori* which possible explanations should be considered. And even for a finite candidate set of causes the question remains to what cause is in effect in each individual case. Each item may be liked or disliked for different reasons and persons may have more than a single interest, different aspects of their taste being relevant for different ratings. In summary, we are looking for a method that would automatically find potential causes, determine which subset of the causes are likely to be relevant for a specific item as well as for a specific person, and in each individual case assign a probability to the fact that a cause will be "active" for a given rating.

Assume for now that the number of potential causes is fixed beforehand, i.e., each variable $z_{u,y}$ may take one out of a finite number of say K values. Then the joint probability of an observation triplet will be given by summing over all possible states of the latent variable,

$$P(u, y, v) = \sum_z P(u, y, v, z) = \sum_z P(v, y|z, u)P(z|u)P(u)\,. \tag{3}$$

We make the crucial independence assumption that conditioned on the true cause z, the vote as well as the selection of the item are independent of the user, implying that $P(v, y|z, u) = P(v, y|z)$. The resulting model can be written as

$$P(v, y|u) = \sum_z P(v, y|z)P(z|u) \tag{4}$$

where we have neglected the user probability $P(u)$. In the first case, where one conditions on both, the user as well as the item, one will get similarly

$$P(v|u, y) = \sum_z P(v|y, z)P(z|u) \tag{5}$$

These parameters have a very intuitive meaning: $P(z|u)$ represents to what extend a user "participates" in a common interest pattern, or more precisely, which fraction of a user's ratings are explained by hidden cause z. $P(v, y|z)$ models which items and item/vote combinations are more likely or less likely to occur in the interest group described by z. Similarly, $P(v|y, z)$ models the probability that a user as a member of interest group z will vote with v on item y.

One can also investigate models in which the role of users and items are reversed, in particular in the symmetric setting of estimating $P(v|u,y)$. Notice that in the typical case where the number of users exceeds the number of items, the dimensionality of the resulting model will be quite different.

The presented approach is closely related to sparse matrix decomposition techniques and the paper effectively extends a model called *Probabilistic Latent Semantic Analysis* which has been used mainly in the context of information retrieval [6]. More details about the conceptional foundations can be found in [5]. A simplified model and preliminary experiments for collaborative filtering have been presented in [7], which mainly dealt with a conceptual framework for latent class models for collaborative filtering.

3.2 Expectation Maximization Algorithm

The Expectation Maximization (EM) algorithm is a standard method for maximum likelihood estimation in latent variable models. The EM algorithm performs two steps in alternation, an E-step, where the expected value for the unobserved variables are computed given the current parameters and a M-step, where these values are used to update the parameters. More details can be found in [5]. In our case the E-step computes the probability that a cause z is associated with an observation triplet $(u, y, v_{u,y})$. By Bayes' rule one obtains

$$P(z|u,y,v_{u,y}) = \frac{P(z|u)\rho(v_{u,y},z,y)}{\sum_{z'} P(z|u)\rho(v_{u,y},z,y)}, \tag{6}$$

where $\rho(v_{u,y},z,y) = P(v_{u,y}|z,y)$ in case (I) and $\rho(v_{u,y},z,y) = P(v_{u,y},y|z)$ in case (II). Notice that the occurrence of a vote on y matters in the computation of the posterior probabilities for z according to (II), but not in case (I).

In the M-step one updates the parameters, i.e., re-estimates the conditional probabilities that define the model. The user participation probabilities are given by

$$P(z|u) = \frac{\sum_{(y,v_{u,y})} P(z|u,y,v_{u,y})}{\sum_{z'}\sum_{(y,v_{u,y})} P(z'|u,y,v_{u,y})} = \frac{\sum_{(y,v_{u,y})} P(z|u,y,v_{u,y})}{|\{(u,y,v_{u,y})\}|}. \tag{7}$$

The update equations for $P(v|y,z)$ and $P(v,y|z)$, respectively, depend on the parametric form that one chooses for those distributions. We assume that the rating scale is quantized, e.g., a rating v might be an integral number between 0 and 5 (as it is the case for the EachMovie data set). In the quantized case one can work directly with multinomial/binomial probability distributions, leading to

$$P(v|y,z) = \frac{\sum_u P(z|u,y,v)}{\sum_{v'}\sum_u P(z|u,y,v')}. \tag{8}$$

Other choices are possible, for example a normal distribution with mean $\mu_{y,z}$ and variance $\sigma^2_{y,z}$ for each combination of items and hidden causes. In this case, the mean would simply be re-estimated in the M-step according to

$$\mu_{y,z} = \frac{\sum_v \sum_u v\, P(z|u,y,v)}{\sum_v \sum_u P(z|u,y,v)}. \tag{9}$$

The Gaussian model has shown slightly worse performance in the experiments, we have therefore focused on the multinomial/binomial model in the experiments.

4 Experimental Evaluation

4.1 Evaluation Criteria

A thorough empirical analysis of collaborative filtering algorithms has been presented in [1] and we have adapted most of the proposed evaluation criteria. The effectiveness of collaborative filtering techniques can be measured in various ways dependent on how the recommender system is used and how results are presented to the user.

In the case that individual items are presented to the user, we use two evaluation metrics. The first score is the absolute deviation of the predicted and actual vote,

$$E_1(u,y) = |p_{u,y} - v_{u,y}|\,. \tag{10}$$

Moreover, for quantized votes, we also evaluate the score

$$E_0(u,y) = \begin{cases} 0, \text{ if } p_{u,y} = v_{u,y} \\ 1, \text{ else.} \end{cases} \tag{11}$$

which only measures whether or not the vote was correctly predicted.

For ranked lists of items we have to assign score to permutations of items. We denote a permutation by τ and the rank of an item y with respect to τ by $\tau(y)$, e.g., the top ranked item y will have $\tau(y) = 1$, the second item $\tau(y) = 2$, and so forth. Items with known votes are not included in the ranking. We then use the following rank score for τ,

$$R(u,\tau) = \sum_y 2^{-\frac{\tau(y)-1}{\alpha-1}} \max(v_{u,y} - \bar{v}, 0), \tag{12}$$

with $\bar{v}$ denoting the overall mean vote. The rationale behind this score is that when presented with a ranked list of items, users will sift through the list starting at the top, until they find a relevant item or simply give up. The probability that a user will ever take notice of an item at rank r is modeled as an exponential distribution with a half-life constant α (set to 4 in our experiments). The total score for a population of users is then measured by (cf. [1])

$$R = 100 \frac{\sum_u R(u,\tau_u)}{\sum_u \max_{\tau'} R(u,\tau')}\,. \tag{13}$$

4.2 Evaluation Protocols

In the evaluation experiments the observed user ratings are partitioned into a training set and a test set. For each user the training data is used to compute votes on unrated items as well as a ranked list of recommendations. For the split into training and test data, we have used a leave-one-out protocol, where we randomly leave out one of the ratings for each user who has at least two observed ratings. This has been called the *All but 1* protocol in [1].

Notice that the above ranking measure takes a very simple form in this case: If the hold-out rating is below the average, simply skip this user. Otherwise, compute the rank of the hold out "positive" item and evaluate the expression corresponding to this item. The maximum achievable score would be to have the test item always be in first place.

4.3 Data Set

Unfortunately, there are very few data sets available to the academic community to test recommender systems. This seems largely because of the potential commercial value of user data and because of privacy issues. The data set we have used in our experiments is the EachMovie data. There are 1623 items in this data set and more than $60,000$ user profiles with a total of over 2.8 million ratings. The rating scale is discrete taking values from 0 to 5.[2]

4.4 Results

We have compared our statistical approach with the memory based approach using the Pearson correlation as a similarity measure. The results on predicting ratings for given items are summarize in Table 1. As one can see, the model-based approach achieves great performance gains in terms of absolute error as well as in terms of prediction accuracy. According to both scores, the model-based approach performs significantly better. The improvement grows with the number of states chosen for the latent class variables, but more or less levels out at around $K = 100$. Notice that we have used early stopping on validation data to avoid overfitting (cf. [5] for more details on complexity control issues).

Although we have not implemented other collaborative filtering techniques like Bayesian clustering or Bayesian networks, the comparison in [1] have demonstrated that memory-based methods achieve excellent results on this benchmark. The results reported in the latter paper on memory-based methods are also in good agreement with the outcome of our experiments. This makes the results in Table 1 even more remarkable. It can also be seen that models with reversed role of users and items show a weaker performance.

The results for generating ranked recommendation lists, the more typical setting for recommender systems are shown in Table 2. Notice that in this case the model denoted by case (II) has been used, which not only predicts the outcome of the rating, but also whether or not a person is familiar with a particular

[2] The original ratings have been multiplied by a factor of 5.

Table 1. Results for predicting individual test ratings on the Each Movie data set. K denotes the number of latent states used to perform the decomposition.

Method	Absolute Deviation	Relative Improvement	Prediction Accuracy	Relative Improvement
Baseline	1.091	0.00	33.4	0.00
Memory-based	0.951	12.8%	35.3	6.3%
Model-based				
K=5	0.972	10.9%	39.3	17.6%
K=10	0.951	12.8%	39.8	19.1%
K=20	0.947	13.2%	39.9	19.5%
K=50	0.937	14.1%	40.1	20.0%
K=100	0.927	15.0%	40.6	21.6%
K=150	0.926	15.1%	40.7	21.9%
K=200	0.924	15.3%	40.8	22.2%
K=400	0.916	16.0%	41.4	24.0%
Model-based (Role of items and users reversed)				
K=20	0.983	9.9%	37.7	12.8%
K=50	0.975	10.6%	38.4	14.9%
K=100	0.973	10.8%	38.5	15.2%

item and/or has rated an item. The achieved performance gains are even more substantial than for predicting ratings of single items. The relative performance gain over the baseline method of ranking items by overall popularity is more than 70%.

4.5 Performance Issues

The advantages of the model-based approach in terms of memory and computation time are also considerable. As far as the memory requirement is concerned, one only needs to store the parameters $P(v, y|z)$ or $P(v, |y, z)$, respectively. The user-specific parameters $P(z|u)$ can be reconstructed as needed. In fact, it is simple to show that the problem of finding $P(z|u)$ reduces to an independent convex cross-entropy minimization problem for each user u. Since the number of users is typically much larger than the number of items, this can make a crucial difference. On the other hand, memory-based approaches suffer greatly from the fact that the data can not be compressed into a model, but has to be kept in main memory at recommendation time.

Similarly there are great savings in terms of computation time. In memory-based approaches, user correlations have typically to be performed on-line which is a very time consuming process. In comparison, the model-based approach only requires to perform of the order of K operations to compute the probability for a single rating or rating/item pair. Since the probabilities $P(z|u)$ are typically very sparse, the actual savings will often be even larger for a slight sacrifice in accuracy. In our experiments, the model-based approach was typically more

Table 2. Rank score results for ordered recommendation lists on the Each Movie data set. K denotes the number of latent states used to perform the decomposition.

Method	Rank Score	Relative Improvement
Baseline	26.95	0.00
Memory-based	36.71	36.2%
Model-based		
K=20	44.64	65.6%
K=50	45.91	70.3%
K=100	45.98	70.6%

than 10-20 times faster compared to the memory-based approach. In practice, memory-based methods often have to subsample the user profile base to allow real-time recommendations for large databases. This is not necessary for our model-based approach.

The main disadvantage of the model-based approach is the computational burden of the model training stage. For the EachMovie data, a typical training run took between 5 and 60 minutes on a Pentium III w/800 MHz (dependent on the dimensionality of the model). Yet, training can be performed off-line where computational resources are typically abundant or at least much less critical.

4.6 Mining User Preferences

Finally, we would like to illustrate that the decomposition of user ratings may lead to the discovery of interesting patterns and regularities that describe user interests as well as dislikes. To that extend we have visualized the items for each latent variable state by sorting them according to the popularity within an interest group as measured by $P(y|z)$ (irrespective of the actual vote). The average vote $\sum_v vP(v|y,z)$ is displayed in rectangular brackets as well. Figure 1 and 2 display the interest groups extracted by the model with $K = 40$, ordered according to the average "positiveness" of each group, computed as $\sum_{y,v} vP(v|y,z)P(y|z)$.

4.7 Conclusion

We have presented a powerful method for collaborative filtering and mining of user data. The method achieves a very good recommendation and prediction accuracy compared to previously proposed methods, in addition it is highly scalable, and extremely flexible. Conceptionally, the decomposition of user preferences is a radically novel idea that clearly distinguishes this approach from traditional memory-based approaches. The use as a data mining tool is another unique benefit of our method.

Acknowledgements. The EachMovie dataset was generously provided by Digital Equipment Corporation. This work was supported in part by the Air Force and the Defense Advanced Research Projects Agency under Grant No. F30602-00-2-0599 and by the National Science Foundation, under Grant No. IIS-0085836.

Fig. 1. EachMovie interest groups number 1-20 out of 40.

Fig. 2. EachMovie interest groups number 21-40 out of 40.

References

1. J. S. Breese, D. Heckerman, and C. Kardie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.
2. D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(2):61–70, 1992.
3. D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwait, and C. Kadie. Dependency networks for density estimation, collaborative filtering, and data visualization. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000.
4. J. Herlocker, Konstanm J., Al Borchers, and J. Riedl. An algorithmic framework for collaborative filtering. In *Proceedings of SIGIR'99*, 1999.
5. T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 289–296, 1999.
6. T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd ACM-SIGIR International Conference on Research and Development in Information Retrieval, Berkeley, California*, pages 50–57, 1999.
7. T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1999.
8. J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to Usenet news. *Communications of the ACM*, 40(3):77–87, 1997.
9. D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, pages 473–480, 2000.
10. P. Resnik, I. Neophytos, M. Suchak, P. Bergstrom, and J. Rield. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM CSCW'94 Conference on Computer-Supported Cooperative Work*, pages 175–186, 1994.
11. U. Shardanand and P. Maes. Social information filtering: Algorithms for automatic 'word of mouth'. In *ACM CHI'95 Conference on Human Factors in Computing Systems*, pages 210–217, 1995.
12. L.H. Ungar and D.P. Foster. Clustering methods for collaborative filtering. In *Workshop on Recommendation Systems at the Fifteenth National Conference on Artificial Intelligence*, 1998.