A Regression-Based Approach for Scaling-Up Personalized Recommender Systems in E-Commerce

Slobodan Vucetic[1] and Zoran Obradovic[1,2]

svucetic@eecs.wsu.edu, zoran@cis.temple.edu

[1]Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752

[2]Center for Information Science and Technology, Temple University, Philadelphia, PA 19122, USA

ABSTRACT

Automated collaborative filtering is one of the key techniques for providing a customization for E-commerce sites. Various neighbor-based recommendation methods are popular choices for collaborative filtering. However, their latency can be a serious drawback for scaling up to a large number of requests that should be processed in real-time. In this paper we propose an alternative regression-based approach that searches for relationships among items instead of looking for similarities among users. Experiments on a movie database provide evidence that the proposed regression-based approach provides significantly better accuracy and is two orders of magnitude faster than the neighbor-based alternatives. Even faster time response with accuracy similar to neighbor-based recommendations was obtained by adjusting the generic recommendations with only the average preference of an active user.

1. INTRODUCTION

In today's society there is an increasing need for automated systems providing personalized recommendations to a user faced with a large number of choices. For example, an increasing choice of available products is caused by companies shifting towards developing customized products that meet specific needs of different groups of customers [11]. The products customization trend coupled with E-commerce where customers were not provided with an option to examine the products "off-shelf" in a traditional sense, make the problem of providing accurate personalized recommendations very important. Increasing the quality of personalized recommendations would increase customer satisfaction and loyalty, at the same time reducing the costs caused by product return. Another example where personalized recommendations are extremely useful is an information overload situation with an amount of data available through Internet and other media largely exceeding the ability of a person even getting a glance of it. Here, automated methods are needed to provide a large number of users with the ability to efficiently locate and retrieve information according to their preferences.

Personalized recommender systems can be classified into two main categories: content-based and collaborative filtering. In content-based filtering, mostly used for retrieving relevant textual documents [6, 12], search is performed for items with content most similar to users' interests. The task of collaborative filtering is to predict preferences of an active user given a database of preferences of other users, where the preferences are typically expressed as numerical evaluation scores. Scores can be obtained explicitly by recording votes from each user on a subset of available items, or implicitly, inferring from a user behavior or reactions regarding a given set of items. Memory-based collaborative filtering algorithms maintain a database of previous users' preferences and perform certain calculations on a database each time a new prediction is needed [3]. The most common representatives are neighbor-based algorithms where a subset of users most similar to an active user is chosen and a weighted average of their scores is used to estimate preferences of an active user on other items [5, 13]. In contrast, model-based algorithms first develop a description model from a database and use it to make predictions for an active user. Published systems of this type include Bayesian clustering, Bayesian networks [3] and classification-based algorithms [2, 9].

Neighbor-based collaborative filtering algorithms are known to be superior to model-based in terms of accuracy [3]. However, their high latency in giving predictions for active users can be a serious drawback in systems with a large number of requests that should be processed in real-time. Also, previous results [4] show that as the number of items evaluated by an active user decreases, the prediction accuracy of neighborhood-based algorithms deteriorates dramatically (demanding an extensive user's effort for successful profiling can be discouraging from using the recommender system). Finally, it is unlikely that two users have exactly the same taste over all possible items while it is more probable that the similarity is larger over certain subsets of items (e.g., two users of a movie recommendation system can share the opinions in dramas while disagreeing in science fiction).

In this paper we propose a regression-based approach to collaborative filtering that searches for similarities between items, builds a collection of experts in the form of simple linear models, and combines them to provide preference predictions for an active user. For example, in a movie recommendation system, if a large positive correlation in votes for movies "12 Monkeys" and "Seven" is identified in the database, a high evaluation score for the movie "12 Monkeys" by an active user implies a higher then average recommendation for the movie "Seven." Linear regression models describing relationship between pairs of movies are used in prediction as local experts and they are integrated using various procedures for combining the experts.

2. BACKGROUND

In this section a more formal description of a recommendation problem is followed by a description of several recommendation algorithms that will be used for comparison with our approach. Performance measures used to compare different algorithms are also introduced in this section.

2.1. Description of the personalized recommendation task

Assuming a database of I items partially evaluated by U users, we are given a U×I matrix $\mathbf{R}$ with element r_{ui} representing an evaluation score of item i by user u. In realistic systems the matrix $\mathbf{R}$ is usually very sparse since users are likely to vote just for a small subset of available items. By r_{u*}, r_{*i}, and r_{**} we denote an average score for each user, an average score for each item, and an overall average score, respectively. Since each user does not vote for each item, by $\mathbf{I}_u$ we denote a subset of items rated by user u. Similar, $\mathbf{U}_i$ denotes a subset of users that evaluated item i. Given the evaluation scores of an active user a on items $\mathbf{I}_a$ the recommender system task is to estimate scores of user a on the remaining items $\mathbf{I}/\mathbf{I}_a$.

2.2. Simple and neighbor-based algorithms

We propose three simple recommendation algorithms establishing accuracy lower bounds on more complex recommendation systems. The first, MEAN, uses the overall average score r_{**} as a score on prediction of any user on any item. The second, GENERIC, uses the average scores r_{*i} of each item i as predictions of a new user's preferences.

The predictions of ADJUSTED_GENERIC for user a are obtained as predictions from GENERIC adjusted by the difference Δr_a between the average score of user a and an average score of an average user over the same set of items, $\mathbf{I}_a$, defined as

$$\Delta r_a = \sum_{i \in \mathbf{I}_a} (r_{ai} - r_{*i}) \,. \tag{1}$$

For neighbor-based algorithms, Pearson correlation is used to measure similarity between user u and active user [4]. Here,

$$w_{a,u} = \frac{\sum_{i \in \mathbf{I}_a \cap \mathbf{I}_u} (r_{ai} - r_{a*})(r_{ui} - r_{u*})}{\sigma_a \sigma_u} \,, \tag{2}$$

where σ_a and σ_u are standard deviations of scores calculated over $\mathbf{I}_a \cap \mathbf{I}_u$. The score of user a for item i is predicted as a weighted sum of the votes of other users p_{ai} computed as,

$$p_{ai} = r_{a*} + \frac{\sum_{u \in \mathbf{U}_i} w_{a,u} (r_{ui} - r_{u*})}{\sum_{u \in \mathbf{U}_i} w_{a,u}} \,. \tag{3}$$

If two unrelated users have a small number of co-rated items, it is probable that their Pearson correlation is high. Existence of such false neighbors can significantly deteriorate the accuracy. Significance weighting [4] is proposed to reduce the weights if the number of co-rated items, n, is smaller than some predetermined number N. If this is the case, the obtained weight is multiplied by n/N. Neighborhood selection is introduced to retain only a small subset of the most similar users for prediction [4]. Predicting an item i can be done effectively by retaining K of the most similar users from $\mathbf{U}_i$. The reported benefits are twofold: computational time needed for each prediction is decreased and slight improvements in accuracy can be observed. Both modifications are implemented in a neighbor-based algorithm used for comparison with the proposed regression-based algorithm. We denote the neighbor-based algorithm with these modifications as NEIGHBOR(N,K), where N and K are adjustable significance and selection parameters of the algorithm.

The bottleneck of neighbor-based algorithms is their on-line speed. To perform a prediction for a new user, Pearson correlations over U existing users should be calculated first using (2), which scales as O(UI), where I is the total number of items. Reducing to the K most similar users for each item to be predicted and applying (3) still scales as O(UI). Also, a whole database **R** of size O(UI) should be maintained for the modified algorithm. Assuming an increasing number of users and items in a database such scaling can be a limitation factor to the practical applications of the neighbor-based algorithms.

2.3. Performance measures

The coverage, mean absolute error and ROC sensitivity were used to compare examined recommendation algorithms. In this study the coverage is being calculated as a percentage of items from a test set for which a given algorithm has been able to give predictions. Mean Average Error (MAE) of predicted scores is a statistical accuracy measure used for collaborative filtering algorithms [4, 13], while ROC sensitivity is a measure of diagnostic power of prediction algorithms [4].

One of the main purposes of recommender systems is to identify items which a new user is likely to prefer. So, a threshold θ_1 is used to obtain a subset of items with predicted scores higher then θ_1. ROC curve is a plot of sensitivity against (1–specificity) of the given predictions. Sensitivity is the probability of accepting a good item, while specificity is the probability of rejecting a bad item. A recommender system can be regarded as successful if its ROC curve is high. As a by-product of ROC curve calculation, one can calculate the accuracy of the recommender system. We denote as ROC(θ_1, θ_2) the accuracy of the system which regards all items with scores above θ_1 as "good items", and declares each prediction above θ_2 as a recommendation.

3. A REGRESSION-BASED COLLABORATIVE FILTERING METHOD

For customized predictions the proposed approach relies on inherent relationships between items instead of looking at similarities between users. Assuming a database consisting of a large number of users voting for most items and given unlimited computational and storage resources, examples from a small neighborhood of an I-dimensional point corresponding to scores of an active user would represent its profile. If an active user provides the scores for items in set $\mathbf{I}_a$ the task is to estimate scores for the set $\mathbf{I}/\mathbf{I}_a$ of non-rated items. In theory, this prediction problem could be approached by learning a nonlinear mapping $f_i(\mathbf{I}_a)$: $\mathsf{R}^{|\mathbf{I}_a|} \rightarrow \mathsf{R}$ and using this function to optimally predict preferences of the active user with respect to the item *i*. An active user can provide scores for one of 2^{I-1} possible subsets of items, and so learning $I2^{I-1}$ functions would optimally solve the recommendation problem. Learning this many functions is computationally unrealistic even for small item sets. In addition, the practical databases are sparse and with insufficient number of users to allow even a small-scale version of the "ideal world" approach.

Therefore, we use a first order approximation of nonlinear mappings $f_i(\mathbf{I}_a)$ as a regression-based approach to collaborative filtering. The first order approximation to predicting score p_{ai} of user *a* on item *i* based on $\mathbf{I}_a$ can be expressed as

$$p_{a,i} = \sum_{j \in \mathbf{I}_a} w_{j,i} \cdot f_{j,i}(r_{aj}), \qquad (4)$$

where $f_{j,i}$ are functions describing the relationship between items *j* and *i*, and $w_{j,i}$ are the corresponding weights. So, assuming a method for choosing the weights $w_{j,i}$ is known, learning I(I–1) one-dimensional functions is sufficient for solving the approximated recommendation problem.

From a different standpoint, function $f_{j,i}$ can be considered as an expert for predicting the score on item *i*, given a score on item *j*. If the active user voted for items from $\mathbf{I}_a$, then for each item from $\mathbf{I}/\mathbf{I}_a$ there are $|\mathbf{I}_a|$ available experts. The recommendation problem can now be approached as an identification of an optimal combination of experts. To make the solution computationally more efficient we model experts as linear functions,

$$f_{j,i}(x) = x\alpha_{j,i} + \beta_{j,i}, \qquad (5)$$

where $\alpha_{j,i}$ and $\beta_{j,i}$ are the only two parameters to be estimated for each expert. The two parameters are estimated using ordinary least squares, and as a by-product we obtain an estimation of the error variance for each expert, $\sigma^2_{j,i}$, which is useful in determining the weights $w_{j,i}$. If the value of $\alpha_{j,i}$ is near zero, this is an indicator that the expert resembles the mean predictor, and that there is no correlation between items *i* and *j*. On the other extreme, if $\alpha_{j,i}$ is close to +1 or –1, the expert is an almost perfect predictor of the score for item *i*.

3.1. Experts integration

Once I(I–1) linear experts are learned, they are combined to give recommendations. One integration method based on simple averaging and two statistically based are proposed in this section.

Average with thresholding

In simple averaging, all $|\mathbf{I}_a|$ experts are given the same weight, $w_{j,i} = 1/|\mathbf{I}_a|$. Since some of the experts can be no better than the mean predictor, they can only deteriorate prediction by decreasing the contribution of good experts. Therefore, simple averaging with thresholding where only good experts are retained is likely to improve predictions. In AVERAGING(θ) we reject all experts whose R-squared value,

$$R^2{}_{j,i} = 1 - \sigma^2{}_{j,i} / \sigma^2{}_i\,, \tag{6}$$

is below θ, where $\sigma^2{}_i$ is variance of all the scores from the database for item *i*. Since some of the experts are not used for prediction, using AVERAGING(θ) for larger θ it is possible that the coverage is incomplete.

Determining optimal weights

Back in 1969, Bates and Granger [1] have suggested that a linear combination of individual predictions can produce the results that are superior to any of the individual predictions. For our problem of combining linear experts, the optimal solution can be derived by minimizing the error variance of the linear combination of experts from (4). The optimal weights $w_{j,i}$ can be found by using $|\mathbf{I}_a| \times |\mathbf{I}_a|$ covariance matrix **C** of prediction errors with elements $\{C_{j,k}\}$ defined as $C_{j,k} = E[e_{j,i} \cdot e_{k,i}]$, where $e_{j,i}$ is the error of expert $f_{j,i}$, $e_{j,i} = \{r_{a,i} - f_{j,i}(r_{a,j})\}$. Weights can be calculated as

$$w_{j,i} = \frac{\sum_{k \in \mathbf{I}_a} \{C^{-1}\}_{j,k}}{\sum_{k,j \in \mathbf{I}_a} \{C^{-1}\}_{j,k}}\,, \tag{7}$$

where $\{C^{-1}\}_{j,k}$ are elements of inverse of **C** [10]. Problems associated with applying (7) include estimating **C**, calculating its inverse when experts are highly correlated and computing this sufficiently fast. To estimate $C_{j,k}$, a subset of users that voted for items *i*, *j* and *k* should be found and errors of $f_{j,i}$ and $f_{k,i}$ should be calculated on this subset. Since a training database is expected to be very sparse, the number of such triples can be too small to properly estimate $C_{j,k}$ even for seemingly large databases. The second problem, as discussed in [8], is that an inverse of C can be unstable for highly correlated experts. Since our linear experts will most often be just slightly better than the mean predictor, they will all be highly correlated. The third problem is computational cost, since the optimal approach requires calculation of an inverse of C which generally takes $O(|\mathbf{I}_a|^3)$ time. Since $|\mathbf{I}_a| = O(I)$ and $|\mathbf{I}/\mathbf{I}_a| = O(I)$, producing predictions for an active user would take up to $O(I^4)$ time. In the following, we propose a computationally efficient approximation of this approach.

Determining suboptimal weights efficiently

One of the main consequences of highly correlated experts is that, from (7), two seemingly identical experts can receive very different weights, the larger being assigned to the slightly more accurate one. To illustrate this, in Table 1 we show the ratio of weights for two experts with error variances $C_{11}=1$ and $C_{22}=1.1$, obtained from (7) when their correlation C_{12} is varied from 0 to 1. Therefore, the effect of increasing C_{12} is similar to the effect of decreasing values of C_{11} and C_{22} by some constant smaller then C_{11}, while $C_{12}=0$. We use this idea for determining weights efficiently.

TABLE 1. The effect of correlation between experts on the optimal weights

C_{12}	0	0.5	0.9	0.95	0.99	1
w_1 / w_2	1.1	1.2	2.0	3.0	11.0	∞

First, we estimate the average correlation ρ between pairs of experts by

$$\begin{aligned} E[(y_1 - y_2)^2] &= E[(y - y_1)^2] - \\ &- 2E[(y - y_1)(y - y_2)] + E[(y - y_2)^2], \end{aligned} \tag{8}$$

where y, y_1 and y_2 are random variables. If we assume that y_1 and y_2 are unbiased experts and *y* is the true value to be predicted, correlation $\rho_{j,k}$ can be estimated from (8) as

$$\rho_{j,k} = \frac{1/2\{\sigma^2{}_{j,i} + \sigma^2{}_{k,i} - E_{D_{j,k}}[(f_{j,i} - f_{k,i})^2]\}}{\sigma_{j,i}\sigma_{k,i}}\,, \tag{9}$$

where $E_{D_{j,k}}[(f_{j,i} - f_{k,i})^2]$ can be calculated over all pairs of scores for items *i* and *j* from the database **R**. We propose to calculate values $\rho_{i,j}$ for several randomly chosen triples (*i*, *j*, *k*) and average them to obtain ρ. Therefore, calculation of ρ, requiring $O(I^2)$ time, needs to be performed just once for the whole database **R**. Then, a new diagonal matrix C^*

$$C^* = diag(C_{j,j}) - \rho \mathbf{I} \cdot \min_j (C_{j,j}) \tag{10}$$

where $diag(C_{j,j})$ is diagonal matrix with elements $C_{j,j}$, and **I** is an identity matrix, should be used instead of C in (7). For the example from Table 1, using both C and C^* results in the same weights when $\rho=\rho_{12}$. Observe that computing the inverse of diagonal matrix C^* requires just O(I) time, and so giving predictions for a new user would require only $O(I^2)$ time. We denote this algorithm as WEIGHTED_AVERAGING.

3.2. Improved regression-based algorithm

One of the problems when dealing with subjective ratings is that two users with similar preferences can have different mean scores for the same subset of items. This notion has been used in ADJUSTED_GENERIC and NEIGHBOR algorithms to adjust the predictions according to each active user's mean score. Here, to improve predictions of WEIGHTED_AVERAGING algorithm we propose an adjustment similar to one used in ADJUSTED_GENERIC. Let us examine two extreme cases to show that the difference Δr_a from (1) cannot be applied directly to WEIGHTED_AVERAGING. The first is the case of experts that can predict the score for item *i* perfectly (without error, $R^2=1$, $\alpha=1$). This prediction does not need to be adjusted by Δr_a since such an expert already describes it. The second is the case of experts that are mean predictors ($R^2=0$, $\alpha=0$). In this case the adjustment is needed, and Δr_a should be added to the prediction. Any realistic combination of experts will be between these two extremes ($0<R^2<1$, $0<\alpha<1$). Therefore, we use the slope α of the best expert as the weighting factor for adjustment. The prediction can now be expressed as

$$p_{a,i} = \Delta r_a \cdot \max_{j \in \mathbf{I}_a} |\alpha_{j,i}| + \sum_{j \in \mathbf{I}_a} w_{j,i} \cdot f_{j,i}(r_{aj}) . \qquad (11)$$

We denote this algorithm as ADJUSTED_WEIGHTED_AVERAGING.

3.3. More about the complexity of the regression based algorithm

Algorithms AVERAGING and WEIGHTED_AVERAGING require $O(I^2)$ time for each new user which is an advantage over O(UI) time needed by NEIGHB algorithm since the number of items is usually significantly lower than the number of users. Effectively, taking into consideration that each new user votes just for a small subset of items, computational time is even lower. If the maximum number of votes for a new user was upper-bounded by a constant, the computational time of regression-based algorithms becomes O(I), while it remains O(UI) for neighbor-based algorithms.

While on-line speed of the proposed regression-based algorithms is clearly superior, they require learning of I(I–1) linear models where O(U) time is required for each. Therefore, the initial learning of experts requires $O(UI^2)$ time which, although does not scale well with the number of items, can be acceptable since it is done off-line. The regression-based algorithms require saving two linear parameters and an estimate of the error variance for each expert, which requires storing 3I(I–1) variables. For most applications this memory requirement compares favorably to memory requirement of neighbor-based algorithms which scales as O(UI).

4. EXPERIMENTAL EVALUATION

In this section we compare collaborative filtering algorithms explained in Sections 2 and 3 on a movie benchmark database. We first describe the database and the evaluation methodology, and later we present a summary of the results.

4.1. The EachMovie database and evaluation methodology

The EachMovie database [7] is a publicly available collaborative filtering benchmark database collected during 18-month period between 1996-1997. It contains ratings from 72,916 users on 1,628 movies with a total of 2,456,676 ratings. Therefore, the matrix is very sparse with only 2.07% rated elements. User ratings were collected on a numeric 6-point scale between 0 and 1, but to make the results more comprehensible we rescaled ratings to integers {0, 1,… , 5}. To perform a number of experiments, and to allow for a fair comparison between different algorithms, we used only a subset of the whole database.

In our experiments the first 10,000 users represented the training set, and the following 10,000 users represented a set of active users. From both sets all users with less then 20 scores were filtered out as well as all the movies receiving less than 50 scores in the training set. This resulted in 3,422 users in the training set and 4,790 ActiveUsers set with 503 retained movies. A relatively low number of training users was desirable for applying neighbor-based methods, while a low number of movies allowed relatively fast learning of 503*502 linear experts needed for regression-based algorithms (trained in 8 hours on a 700MHz NT-based computer with 256MB memory).

In the first set of experiments we have randomly selected five votes from each active user for testing, while the rest was used as the set of training scores $\mathbf{I}_a$. We call this protocol *AllBut5*. Therefore, 5*4,790 predictions were made with each algorithm on the same testing set, and these results were used to measure MAE, ROC curve and the coverage of different algorithms.

In accordance with [3] we performed another set of experiments allowing a smaller number of active user scores. Here, we randomly selected 2, 5, or 10 votes from each active user as the observed scores, and then predicted remaining scores. The three protocols were named *Given2*, *Given5*, and *Given10*. Such a small number of scores given by active user is likely to be the environment for realistic recommender systems.

4.2. Summary of results

In Table 2 we report the results for different collaborative filtering algorithms. We used ROC(θ_1 = 4,θ_2 = 3.5) to report the classification accuracy. Therefore, each item with a score of 4 or 5 was regarded as a "good" item, and each prediction above 3.5 was regarded as the recommendation. Out from several considered choices of parameters K and N, we show performance results for the best performing neighbor-based algorithm with parameters K=80 and N=50. As could be seen it had an almost complete coverage, but with accuracy just slightly better then ADJUSTED_GENERIC algorithm. AVERAGING algorithms showed large sensitivity to the threshold, with θ = 0.06 offering the best compromise between accuracy and coverage. Accuracy for θ = 0.10 was the best, but with the coverage of only 84.5%.

For WEIGHTED_AVERAGING algorithm, it was determined that an average correlation between experts was slightly larger then 0.9, and this value was used as the default value for equation (10). For illustration, we also report the results for ρ = 0.5 and ρ = 0.95. It could be seen that the value of ρ = 0.95 improved results slightly, effectively meaning that it is desirable to assign an even higher weight to the best experts. Since the coverage of WEIGHTED_AVERAGING was 100%, it can be concluded that, overall, it is slightly more successful than ADJUSTED_GENERIC, NEIGHBOR and AVERAGING algorithms.

Algorithm ADJUSTED_WEIGHTED_AVERAGING was clearly superior to other algorithms, indicating that an introduced adjustment can significantly boost the prediction accuracy. It should be noted that a similar adjustment can be incorporated to AVERAGING algorithm and that similar improvements could be expected. In Figure 1 we show ROC curves for NEIGHBOR (K=80, N=50) and REG_WA(ρ = 0.9) algorithms, and it could be seen that ADJUSTED_WEIGHTED_AVERAGING achieved better accuracy over the whole curve. Also, in Figure 2 we plot the MAE of NEIGHBOR and ADJUSTED_WEIGHTED_AVERAGING algorithms for 4 categories of users depending on the number of given scores, $|\mathbf{I}_a|$. As could be noted, while NEIGHBOR was very sensitive to $|\mathbf{I}_a|$, showing the lowest accuracy for the active users giving less then 25 votes, ADJUSTED_WEIGHTED_AVERAGING was just moderately sensitive indicating that it can be successfully used even for users providing just a few votes.

TABLE 2. Prediction results of different algorithms

Algorithm	MAE	ROC(4,3.5) accuracy	Coverage [%]	Elapsed time [s]
MEAN	1.054	-	-	-
GENERIC	0.925	0.652	100	1.1
ADJUSTED_GENERIC	0.853	0.697	100	2.9
NEIGHBOR(K=80,N=50)	0.848	0.698	99.9	38056
AVERAGING (θ = 0.00)	0.883	0.681	100	68
AVERAGING (θ = 0.06)	0.850	0.702	98.1	
AVERAGING (θ = 0.10)	0.842	0.706	84.5	
WEIGHTED_AVERAGING(ρ = 0.9)	0.844	0.705	100	70
WEIGHTED_AVERAGING (ρ = 0.5)	0.849	0.703	100	
WEIGHTED_AVERAGING (ρ = 0.95)	0.842	0.707	100	
ADJUSTED_WEIGHTED_AVERAGING (ρ = 0.9)	0.818	0.716	100	73
ADJUSTED_WEIGHTED_AVERAGING (ρ = 0.5)	0.823	0.713	100	
ADJUSTED_WEIGHTED_AVERAGING (ρ = 0.95)	0.817	0.717	100	

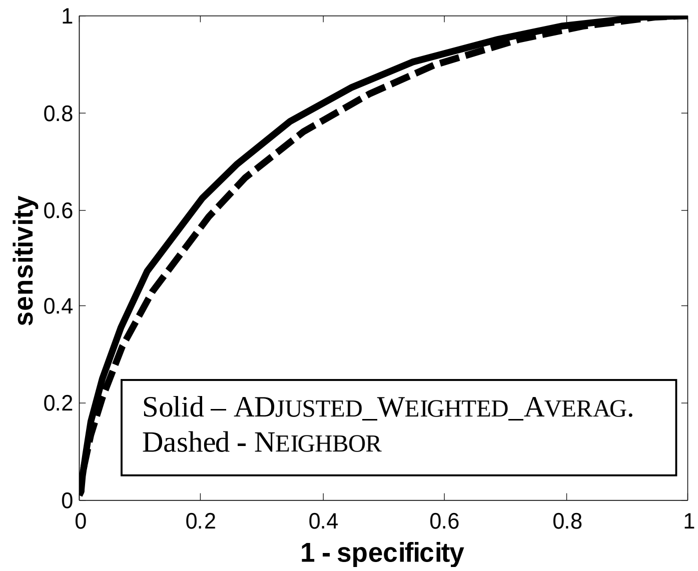

Figure 1. ROC curves for NEIGHBOR and ADJUSTED_WEIGHTED_AVERAGING algorithms

As explained in previous two sections, regression-based algorithms have superior on-line speed to neighbor based algorithms. Our implementation of these algorithms in Matlab on a 700MHz NT-based computer with 256MB memory showed that when performing 5*4,790 predictions ADJUSTED_WEIGHTED_AVERAGING was 521 times faster then NEIGHBOR, and that ADJUSTED_WEIGHTED_AVERAGING was 25 times slower then ADJUSTED_GENERIC (Table 2). Although we do not claim our implementation is optimal, these results validate results from the analysis of on-line speed of neighbor-based and the proposed regression-based algorithms.

Comparison of 4 different protocols is presented in Table 3 where *AllBut5* are results taken from Table 2. As could be seen, the regression-based approach is very robust to the number of scores given by an active user. It is also evident that performances of ADJUSTED_GENERIC and ADJUSTED_WEIGHTED_AVERAGING deteriorated significantly for *Given5* and *Given2* protocols. This was to be expected since estimates of adjustment Δr_a using only 2 or 5 scores have extremely low confidence. Our work in progress is aimed towards using statistically based pessimistic estimates of adjustments for improving the performance of both methods.

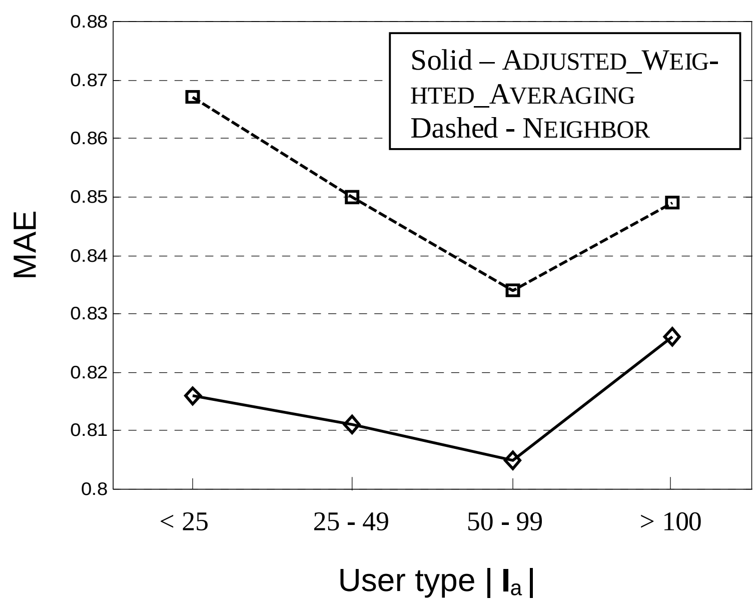

Figure 2. Dependence of MAE on the number of scores | I_a | given by the active user

TABLE 3. MAE of different protocols

Algorithm	*AllBut5*	*Given10*	*Given5*	*Given2*
MEAN	1.054	1.066	1.066	1.067
GENERIC	0.925	0.935	0.935	0.937
ADJUSTED_GENERIC	0.853	0.883	0.925	1.023
NEIGHBOR(K=80,N=50)	0.848	0.912	0.991	1.195
WEIGHTED_AVERAGING(ρ = 0.9)	0.844	0.882	**0.894**	**0.906**
ADJUSTED_WEIGHTED_AVERAGING (ρ = 0.9)	**0.818**	**0.861**	0.901	1.018

5. CONCLUSIONS AND FUTURE WORK

Automated collaborative filtering is one of the key techniques for providing a customization for E-commerce sites. Various neighbor-based recommendation methods are popular choices for collaborative filtering. However, their latency can be a serious drawback for scaling up to a large number of requests that should be processed in real-time. In this paper we proposed a regression-based approach to collaborative filtering that searches for similarities between items, builds a collection of experts in the form of simple linear models, and combines them in a proper way to give predictions for a new user. We examined a number of procedures for combining the experts, ranging from simple averages to statistically sound ones. In addition, we proposed three simple algorithms for comparison with neighbor and regression-based algorithms. One of them, the ADJUSTED_GENERIC, showed as very robust and difficult to outperform despite its simplicity. Experiments on a movie database provided evidence that, while providing maximal coverage, the proposed regression-based approach provides significantly better accuracy and is two orders of magnitude faster than the neighbor-based alternatives.

However, the overall accuracy improvements over simpler prediction algorithms were still narrow, and the goal of further research is to search for further accuracy improvements. There are several directions for improving the accuracy, speed and memory requirements of the regression-based algorithm. One of the obvious extensions of our approach is to use more complex experts instead of linear ones. To improve the accuracy, combining predictions of different types of known algorithms can also be a viable approach if computational time is of no concern. To improve the on-line speed and memory requirements of the proposed procedure, deleting experts with poor predicting capabilities can be an acceptable alternative. If there are no available experts to predict on a given item, maximum coverage could be saved by using simpler models such as ADJUSTED_GENERIC. With such an approach, the accuracy would not significantly deteriorate since deleted experts are not much better than the mean predictor.

Since the error variance of each expert is estimated as a part of the regression-based algorithms, this knowledge can possibly be used for guided on-line recommender systems where, based on the previous votes, an active user is asked to vote on the items that would maximally decrease overall prediction error. Deriving an optimal procedure for guided voting is the topic of our research in progress.

6. REFERENCES

[1] Bates, J. M., and Granger, C. W. J. (1969), The combination of forecasts, *Operational Research Quarterly*, 20, pp. 451-468, 1969.

[2] Billsus, Daniel and Pazzani, Michael J. (1998), Learning collaborative information filters, *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 46-54, 1998.

[3] Breese, John S., Heckerman, David and Kadie, Carl (1998), Empirical analysis of predictive algorithms for collaborative filtering, *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 43-52, 1998.

[4] Herlocker, Jonathan L., Konstan, Joseph A., Borchers, Al and Riedl, John (1999), An algorithmic framework for performing collaborative filtering. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 230-237, 1999.

[5] Konstan, Joseph A., Miller, Bradley N., Maltz, David, Herlocker, Jonathan L., Gordon, Lee R., and Riedl, John (1997), GroupLens: applying

collaborative filtering to Usenet news, *Communications of the ACM*, 40(3), pp. 77-87, 1997.

[6] Maes, Pattie (1994), Agents that reduce work and information overload, *Communications of the ACM*, 37(7), pp. 30-40, 1994.

[7] McJones, Paul (1997), EachMovie collaborative filtering data set, DEC Systems Research Center, http://www.research.digital.com/SRC/eachmovie/, 1997

[8] Merz, Christopher J. and Pazzani, Michael J. (1999), A principal components approach to combining regression estimates, *Machine Learning*, 36(1-2), pp. 9-32, 1999.

[9] Nakamura, Atsuyoshi and Abe, Naoki (1998), Collaborative filtering using weighted majority prediction algorithms, *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 395-403, 1998.

[10] Newbold, P., and Granger, C. W. J. (1974), Experience with forecasting univariate time series and the combination of forecasts, *Journal of the Royal Statistical Society, ser. A*, 137, pp. 131-146, 1974.

[11] Pine, B. Joseph (1993), *Mass Customization*, Harvard Business School Press, Boston, MA, 1993.

[12] Salton, Gerard and Buckley, Christopher (1988), Term-weighting approaches in automatic text retrieval, *Information processing & Management*, 24(5), pp. 513-523, 1998.

[13] Shardanand, Upendra and Maes, Pattie (1995), Social information filtering: algorithms for automating "word of mouth", *Proceedings of Computer Human Interaction*, pp. 210-217, 1995.