Online Queries for Collaborative Filtering

Craig Boutilier and Richard S. Zemel
Department of Computer Science
University of Toronto
Toronto, ON, M5S 3H5, CANADA
{cebly,zemel}@cs.toronto.edu

Abstract

Collaborative filtering allows the preferences of multiple users to be pooled in a principled way in order to make recommendations about products, services or information unseen by a specific user. We consider here the problem of online and interactive collaborative filtering: given the current ratings and recommendations associated with a user, what queries (new ratings) would most improve the quality of the recommendations made? This can be cast in a straightforward fashion in terms of *expected value of information*; but the online computational cost of computing optimal queries is prohibitive. We show how offline precomputation of bounds on value of information, and of prototypes in query space, can be used to dramatically reduce the required online computation. The framework we develop is quite general, but we derive detailed bounds for the multiple-cause vector quantization model, and empirically demonstrate the value of our active approach using this model.

1 Introduction

Collaborative filtering (CF) has attracted considerable attention over the past decade due to the ease of online data accumulation and the pressing need in many applications to make suggestions or recommendations to users about products, services or information. When other users have viewed (say) a product of interest and offered ratings of that product, the existing ratings can be used to predict the rating of a subject who has not seen the product. Specifically, if users with similar "interests" to the subject (as determined using ratings by the subject on other products) have rated the product in a particular way, we might want to recommend that product to our subject. In this way, collaborative filtering allows the preferences of multiple users to be pooled in a principled way in order to make recommendations. The collaborative filtering approach forms the basis of many recommender systems [Breese *et al.*, 1998; Konstan *et al.*, 1997; Nguyen and Haddawy, 1998; Hofmann and Puzicha, 1999; Goldberg *et al.*, 2000], applied to areas as diverse as books, movies, jokes, and newsgroup articles.

A number of different approaches to collaborative filtering have been proposed, including correlation analysis [Konstan *et al.*, 1997], naive Bayes classifiers [Breese *et al.*, 1998], latent class models [Hofmann and Puzicha, 1999], and PCA [Goldberg *et al.*, 2000]. Many of these approaches construct explicit probabilistic models of the domain, positing features or clusters of users and/or products, and relating user and product features to predicted ratings. In many of these models, reasonable results have been obtained.

It is natural to ask in such settings whether additional ratings provided by a user can increase the quality of recommendations made for that user (or equivalently, increase the accuracy of our predicted ratings). Specifically, suppose a user has rated k products, on the basis of which we make predictions for her ratings of the unrated products. If we have the opportunity to ask the user for a rating of a $k+1^{st}$ product, we want to know whether: (a) this new rating can improve our predictions (and ultimately the value of the recommendation we make); and (b) which product offers the greatest expected benefit in this regard. An *active* approach to collaborative filtering involves asking queries of this type when the expected benefit outweighs the cost (e.g., delay, bandwidth, or cognitive burden) associated with the query.

Approaches to CF that learn explicit probabilistic models of the domain facilitate the analysis of this problem: we can pose it in terms of *expected value of information (EVOI)*. Prior to asking a query, we have a distribution over the ratings of unrated products. We assume some decision criterion used to make recommendations based on this prior, as well as a measure of the expected utility of any decision. If we query the user about an unrated product q and obtain a rating r in response, the posterior over ratings will generally lead to a different recommendation with different expected utility. Taking expectation of these utilities with respect to possible responses, we obtain the *myopic* (i.e., single-step lookahead) EVOI for query q. The query with maximum EVOI is most appropriate, so long as its value exceeds the cost of the query.[1] This model can be especially useful when dealing with new users, or users who have not yet populated rating space sufficiently. This approach allows maximum benefit to be derived from fewer product ratings. It is also useful in settings in which we have low confidence in our predicted ratings. In such settings, the benefit of having a user rate several unseen products (e.g., by playing a music or movie clip) before making a recommendation may outweight the costs.

Unfortunately, computing (myopic) EVOI exactly is computationally difficult. In principle, we could ask a user about any unrated product, and for each possible response (rating), we must generally compute the posterior over the remaining ratings to determine the new optimal decision. This requires $O(M^2\rho)$ posterior computations, where M is the number of products and ρ the number of ratings. Worse yet, this computation must be performed online, while interacting with the user. Since CF is most useful in situations with large numbers of users and products, this in unlikely to be feasible except in the most trivial settings.

We consider approaches that allow us to bound the expected changes in these posteriors in a user-independent fashion. By constructing such bounds offline (using the learned model), we can dramatically reduce the number of online posterior computations needed to determine the query with maximum EVOI. In addition, we can use properties of the learned model to construct a small set of *prototype* queries, further reducing the online computational complexity, with only a small sacrifice in decision quality. The framework we develop is quite generic, and can be applied to any CF algorithm that produces an explicit probabilistic model of the domain. However, the details will depend on the specifics of the model in question. Here we develop these details for the specific case of the *multiple-cause vector quantization (MCVQ)* model developed by Ross and Zemel [2002]. However, the development will be similar for most other common types of probabilistic models used for CF.

The remainder of the paper is organized as follows. In Section 2, we discuss collaborative filtering and the MCVQ model. Section 3 describes value of information in general terms, and spells out the details the specific case of the MCVQ model. We show empirically that supplementing product ratings using myopic EVOI in the MCVQ model descreases loss more quickly than adding random ratings. Section 4 details a method for bounding the impact a query can have on the mean rating of a target product in a user-independent (offline) fashion, allowing the query with maximum EVOI to be computed more effectively online. Empirical results again demonstrate a significant amount of pruning can be obtained in the MCVQ model. We discuss some preliminary ideas pertaining to offline prototyping of queries in Section 5, which further reduces the space of queries one needs to consider. We conclude with some suggestions for refinements to the model and directions for future research.

The notion of active collaborative filtering has been suggested by Pennock and Horvitz [2000]; but this work does not suggest specific techniques for implementing the active component in the face of the intensive online computational challenges facing any use of EVOI. Our work is also related to more generic forms of active learning (e.g., [Cohn *et al.*, 1996]), though our focus is on the more specific details of CF and ensuring that online computation is tractable.

2 Collaborative Filtering

We begin by establishing notation and basic background on collaborative filtering. We then describe the MCVQ model.

2.1 The Collaborative Filtering Problem

The basic task in collaborative filtering is to predict the utility of items to the target or active user based on a database of ratings from a population of other users. Ratings can be classified as either explicit or implicit. Explicit rating refers to a user directly specifying his/her preference for an item (e.g., GroupLens users rated each Netnews article on a scale of one (bad) to five (good) [Konstan *et al.*, 1997]). Implicit rating entails interpreting user behavior or selections, for example based on browsing data in web applications, purchase history, or other types of information access patterns. We focus here on applications in which the rating database contains explicit ratings.

From a probabilistic perspective, the aim is to estimate the probability that the active user will assign a particular rating to an as-yet unobserved item. The basic paradigm in CF is that offline processing on the training set of user ratings produces model parameter values, which permit the online estimation of these probabilities based on the set of items for which the active user has provided ratings. Batches of user data can also be used to update the parameter values.

Let P denote the distribution over rating vectors for a generic CF model, trained on existing data. Let $knwn(i)$ denote the set of products for which user i has provided ratings, with $\mathbf{r}_i^{knwn}$ denoting vector of ratings over this set. Let $unkn(i) = M \setminus knwn(i)$. From this we obtain a posterior distribution for each $j \in unkn(i)$:

$$P_w(R_{ij}) = P(R_{ij}|\mathbf{r}_i^{knwn}) \tag{1}$$

where $w = |knwn(i)|$. In general, we use the w subscript to denote posterior distributions that take into account the

[1] This myopic approximation of EVOI is generalized below. Our focus in this paper is on myopic approaches, however.

w known ratings of the active user. Note that R_{ij} can be treated as either a discrete or continous variable.

2.2 Probabilistic Models

Original statistical collaborative filtering approaches predicted unobserved ratings by weighted linear combinations of other users' ratings, with weights derived from the correlation between each user and the active user [Konstan *et al.*, 1997].

Latent factor models have also been applied to this problem. A simple form of these, a mixture or vector quantization (VQ) model, assumes that users cluster into classes with common tastes and preferences. In the standard naive Bayes formulation, the ratings of items j are conditionally independent, given the class of user i:

$$P(C_i = c, r_1, ...r_M) = P(C_i = c) \prod_{j=1}^{M} P(R_{ij} = r_j | C_i = c)$$

The parameters of the model—the probabilities of class membership $P(C_i = c)$, and the conditional probabilities $P(R_{ij} = r_j | C_i = c)$—are estimated offline. Online processing simply re-estimates the class membership probabilities based on the observed ratings and uses these to refine the posterior over ratings of unobserved items:[2]

$$P_w(R_{ij}) = \sum_c P(R_{ij} = r_j | C_i = c) P(C_i = c | \mathbf{r}_i^{knwn})$$

A second form of latent factor model is the aspect model [Hofmann and Puzicha, 1999], which associates an unobserved class variable, the aspect z, with each observation. Unlike the VQ model, each observation here consists of pair, an item j and a user i. The key assumption is that i and j are independent, conditioned on z:

$$P(i, j) = \sum_z P(j|z) P(z|i) P(i)$$

Each aspect implies a distribution over items and ratings, and each user is modeled as a convex combination of aspects. This model offers more flexibility than the VQ model, in that a user can be described by several aspects, since $P(z|i)$ serve as the mixture weights of the aspects. However, the aspect model is not a proper generative model of input vectors, since i is a dummy index referring to the list of users in the training set. This variable has as many possible values as there are training users so the model learns $P(z|i)$ only for those users, and there is no natural way to examine the probability of some unseen user.

[2]Note that unobserved items are treated as missing-at-random. In many cases this assumption is not true, as the fact that an item is unobserved can be telling. However, we make this simplifying assumption for all models considered in this paper, even though each can be elaborated to handle this additional information.

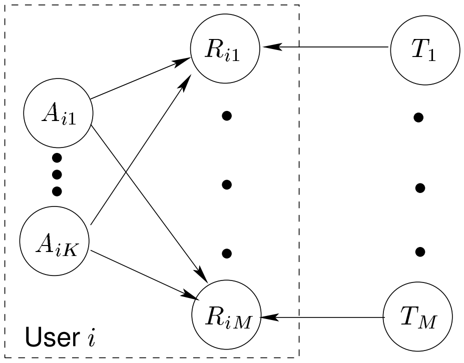

Figure 1: Graphical model for the MCVQ model. Circles denote random variables and the dashed rectangle shows the plate (i.e., repetitions) over the data (users).

2.3 Multiple-Cause Vector Quantization

MCVQ is a new probabilistic model for unsupervised learning which is particularly relevant to CF. The key assumption is that dimensions of the data can be separated into several disjoint subsets, or *multiple causes*, which take on values independently of each other. We also assume each cause is a *vector quantizer*, i.e., a multinomial with a small number of discrete states. Given a set of training examples, the MCVQ model learns the association of data dimensions with causes, as well as the states of each VQ.

In the context of CF, the causes could correspond to *types* of items or products, and the states of a particular type could correspond to a user's *attitudes* or rating profiles that a user can adopt towards items of the given type. In a music rating database, for example, each piece of music could be considered as a mixture of types or genres, and a user can be described as a mixture of attitudes towards each type, where each attitude implies a particular distribution over ratings for each piece of that type. In different terms, a particular user can be described as a *composite sketch*: a selection of the attitudes towards each type.

The notation and basic equations of MCVQ are as follows. Each item j is one of K types, or VQs: $P(T_j = k), k \in \{1, ..., K\}$. Corresponding to each type k there are L different attitudes that a user can adopt: $P(A_{ik} = l), l \in \{1, .., L\}$. Distributions over ratings can be estimated given these two quantities: $\theta^r_{jkl} \equiv P(R_{ij} = r | T_j = k, A_{ik} = l)$, and the posterior over an item's rating is:

$$P_w(R_{ij} = r) = \sum_k P(T_j = k) \sum_l P(A_{ik} = l | \mathbf{r}_i^{knwn}) \theta^r_{jkl}$$

This posterior computation uses the model parameters that are estimated offline from the large ratings database: $P(T_j = k)$, θ^r_{jkl}, and $P(A_{ij} = l)$. The only online computation in the model entails updating the attitude distributions as more item ratings are observed:

$$
\begin{aligned}
P_w(A_{ij} = l) &\equiv P(A_{ij} = l | \mathbf{r}_i^{knwn}) \\
&= \alpha \prod_{h \in knwn(i)} [\sum_{k' \neq k} P(T_h = k') \sum_{l'} P(A_{ik'} = l') \theta^{r_h}_{hk'l'} \\
&\qquad + P(T_h = k) \theta^{r_h}_{hkl}] P(A_{ik} = l)
\end{aligned}
$$

where α is a normalizing constant.[3]

A variational EM algorithm is used to learn the model parameters, and infer hidden variables (attitude distributions) given observations. Details of learning and inference in the model can be found in [Ross and Zemel, 2002].

An example application of MCVQ to CF involves the EachMovie dataset; this is the target database for the experiments described in this paper. The dataset contains ratings, on a scale from 1 to 6, of a set of 1649 movies, by 74,424 users. We divide the full dataset into two subsets: The **main** subset includes users who rated at least 75 movies and movies rated by at least 126 users, leaving a total of 1003 movies and 5831 users. The **sparse** subset includes the rest of the users. We train the model on the **main** subset, and test it on both. We further split the **main** dataset randomly into 1000 users in a test set, leaving 4831 users in the training set. We ran MCVQ with 12 VQs and 4 components per VQ on this dataset.[4] An example of the results, after 15 iterations of EM, is shown in Fig. 2.

MCVQ resembles the aspect model in that a given user can be represented by a multitude of hidden factors, but it is a proper generative model, in that a novel user's rating vectors can be generated by sampling from the distribution over types for each item, and sampling from the attitudes over each type, and then sampling from the combined rating distribution. Also note that an MCVQ model with a single type or VQ is equivalent to the standard VQ model described above. The representational scheme in MCVQ is powerful due to its combinatorial nature: while the standard VQ containing N components can represent at most N items, if we divide the N into j N/j-component VQs, MCVQ can represent $j^{N/j}$ items.

VQ 2	VQ 6
The Shawshank Redemption 5.5 (5) **Taxi Driver** 5.3 (6) **Dead Man Walking** 5.1 (-) Billy Madison 3.2 (-) Clerks 3.0 (4) Forrest Gump 2.7 (2)	**The Godfather** 5.8 (6) **Pulp Fiction** 5.7 (5) **Get Shorty** 5.2 (-) Sound of Music 2.9 (2) Lawrence of Arabia 2.6 (3) Mary Poppins 2.4 (1)
Sling Blade 5.4 (5) **One Flew ... Cuckoo's Nest** 5.3 (6) **Dr. Strangelove** 5.2 (5) The Beverly Hillbillies 2.0 (-) Canadian Bacon 1.9 (4) Mrs. Doubtfire 1.7 (-)	**Mary Poppins** 5.3 (5) **The Wrong Trousers** 5.2 (6) **Willy Wonka** 5.0 (6) Married to the Mob (3.3) 4 Pulp Fiction 3.2 (2) GoodFellas 2.9 (2)

Figure 2: The MCVQ representation of two test users in the EachMovie dataset. The 3 most conspicuously high-rated (bold) and low-rated movies by the most active states (dominant attitudes) of 2 of the 12 VQs are shown, where conspicuousness is the deviation from the mean rating for a given movie. Each state's predictions, $\bar{\theta}_{jkl}$, can be compared to the test user's true ratings (in parentheses); the model's prediction is a convex combination of state predictions. Note the intuitive decomposition of movies into separate VQs, and that different states within a VQ may predict very different rating patterns for the same movies.

3 Myopic EVOI in Collaborative Filtering

We first review the basics of value of information in a generic CF context, and then derive the details of EVOI computations in the MCVQ model, demonstrating that actively generated queries (data points) with high EVOI provide better results than randomly generated queries.

3.1 Value of Information

Most models of CF produce an explicit probabilistic model of the domain, giving rise to distributions over ratings for a specific user-product pair based on attributes of the user and product in question. The MCVQ model described above, for example, can be seen as producing a distribution over types for each product, a distribution over user attitudes towards products of each type, and a distribution over the ratings of product j by user i conditioned on their respective types and attitudes.

For simplicity, we assume that the system can make recommendations only for a single product, and that the utility of any recommendation is given by its actual rating.[5] Thus the recommendation with highest expected utility is that product with highest mean rating. We define the *value* of $P_w(R_{ij})$ to be

$$V(P_w) = \max_{j \in unkn(i)} \sum_r r \cdot P_w(R_{ij} = r)$$

If we ask user i to rate product $q \in unkn(i)$ and obtain response r_q, our new posterior over ratings is $P_w^{r_q}$, with value defined as:

$$V(P_w^{r_q}) = \max_{j \in unkn(i) \setminus \{q\}} \sum_r r \cdot P_w^{r_q}(R_{ij} = r)$$

The (myopic) *expected value of information* associated with query q is the expected *improvement* in decision quality one obtains after asking q:

$$EVOI(q, P_w) = \sum_{r_q} \left(P_w(R_{iq} = r_q) V(P_w^{r_q}) \right) - V(P_w) \quad (2)$$

[3]In order to reduce the number of parameters in the model, ratings are treated as continuous variables, so each state l of type k has a mean $\bar{\theta}_{jkl}$ and variance σ^2_{jkl} in its rating predictions for item j. These are converted into multinomial distributions over ratings through binning and normalization.

[4]We use this same trained MCVQ model throughout the paper. The model performance varies somewhat for different numbers of VQs and components per VQ, but this variation is not the central focus of this paper.

[5]Other decision criteria can be used.

The myopic EVOI approach to active collaborative filtering requires that we ask that query whose EVOI is maximal, as long as it is positive, or above some "query cost" threshold.

It is important to note that this myopic approximation to true expected value of information can be led astray. For instance, if two queries could lead to a dramatic shift in our ratings prediction for a user, but neither query individually has any effect, myopic EVOI will be unable to discover this potentially valuable pair of queries. Solutions to this problem include using multistage lookahead, or more accurately, modeling the entire interactive process as a sequential decision problem. We leave the study of these more computationally demanding approaches to future work.

3.2 EVOI in the MCVQ Model

The computations involved in computing myopic EVOI in the MCVQ model are reasonably straightforward. We develop these in this section, but emphasize that the application of EVOI to other CF models would proceed in an analogous fashion. We assume M products, K types (or VQs), L user attitudes toward products of a specific type (or components), and rating set $\{1, ..., \rho\}$. We assume a trained MCVQ model with parameters: $P(T_j = k)$, for $j \leq M, k \leq K$; $P(A_{ik} = l)$, for $k \leq K, l \leq L$; and $P(R_{ij} = r|T_j = k, A_{ik} = l) = \theta^r_{jkl}$, for $j \leq M, k \leq K, l \leq L, r \leq \rho$. Note that the parameters θ^r_{jkl} are independent of the user i, and that R_{ij} is independent of $A_{ik'}$ given $T_j = k$, for any $k' \neq k$.

Expected value of information can be computed in the fashion described above in the MCVQ model. The specifics of the MCVQ model dictate only how to update ratings distributions given a response to a query. Assume a user i has provided response r_q to query q. We then compute the posterior for any $j \in \mathit{unkn}(i) \setminus q$:

$$\begin{aligned}
P_w^{r_q} &= P(R_{ij} = r|R_{iq} = r_q, \mathbf{r}_i^{knwn}) \\
&= \sum_{k \leq K} P(T_j = k) \sum_{l \leq L} \theta^r_{jkl} P_w(A_{ik} = l|R_{iq} = r_q) \\
&= \sum_{k \leq K} P(T_j = k) \sum_{l \leq L} \theta^r_{jkl} \alpha [\sum_{k' \neq k} \{P(T_q = k') \\
&\quad \sum_{l' \leq L} P_w(A_{ik'} = l')\theta^{r_q}_{qk'l'}\} + P(T_q = k)\theta^{r_q}_{qkl}] P_w(A_{ik} = l)
\end{aligned}$$

Given these posterior calculations we can compute EVOI of any query using Eq. 2 above.

We evaluate the efficacy of this approach empirically by examining the change in *model loss* for the MCVQ model as we update ratings based on responses to queries. Model loss is defined as the difference between the user's actual utility (rating) for the best item we could have recommended and the actual utility for the item recommended by the model. The model recommendation is the item with highest mean rating (i.e., the item *predicted* to be best). In this experiment, we fix w, the number of observed ratings, and randomly select the items to be observed for each test user, holding out ratings of other items by this user. We then compute the model loss for those observations by subtracting the user's true rating of the model's highest ranked held-out item (predicted utility) from the user's rating of her highest-ranked held-out item (highest utility). We evaluate the *change* in model loss due to a query q by observing the rating of item q, updating the model, and comparing the loss of the prior and posterior model (where the posterior model loss is defined over the reduced set of held-out items). We compare the change in model loss using the query with maximum EVOI with that obtained using random queries.[6]

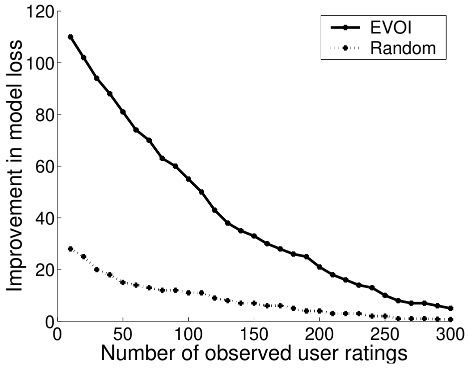

Figure 3: The total improvement in model loss (difference between actual and predicted utility) for MCVQ for varying number of observed ratings of test users. For each test user, the maximum improvement is 5 (ratings range from 1-6), and the total improvement sums the improvement across the set of 1000 test users. Each datapoint is an average of 5 test runs for each test user, with a random selection of observed ratings on each run.

Fig. 3 shows that selecting the held-out item to query (or observe) based on EVOI leads to significantly greater improvements in model loss than a random selection strategy, particularly for small values of w. This dependence on w conforms with the intuition that the value of information should decrease with increasing knowledge of the user, as the posterior over ratings stabilizes. In fact, we could use a threshold on EVOI as a form of "query cost", so that if the maximum EVOI value does not exceed the threshold, the system would not query the user. Instead, it would make a recommendation to the user. We note that these results involve testing users drawn from the **main** data set. Results using the **sparse** user set are qualitatively similar.

4 Bounding Mean Rating Change

The straightforward computation of the EVOI of a query q in the MCVQ model requires $O(\rho M)$ posterior com-

[6]Queries are restricted to held-out items, since these are the only queries for which can can obtain actual "responses."

putations. Since each unrated product is a potential query, determining the query with maximum EVOI requires $O(\rho M^2)$ posterior calculations. Since this process must be engaged online, while interacting with the user, this approach to active collaborative filtering is unlikely to be feasible.

Fortunately, we can reduce the number of posterior calculations by bounding the impact a specific rating associated with product q can have on the mean rating of product j. We do this in a user independent fashion, allowing the computation of these bounds offline (e.g., at the same time a new model is being learned with a new batch of data). As before, we assume a learned MCVQ model. We proceed in several stages.

We first bound the difference in the posterior probability of a rating $P_w^{r_q}(R_{ij} = r) = P(R_{ij} = r | R_{iq} = r_q, \mathbf{r}_i^{knwn})$ given response r_q to query q and the prior $P_w(R_{ij} = r)$. We have

$$\begin{aligned} & P_w^{r_q}(R_{ij} = r) - P_w(R_{ij} = r) \\ & \quad = \sum_{k \le K} P(T_j = k) \sum_{l \le L} [P_w^{r_q}(A_{ik} = l) - P_w(A_{ik} = l)] \theta_{jkl}^r \\ & \quad \le \sum_{k \le K} P(T_j = k) \sum_{l \le L} \Delta_{kl}^{q r_q} \theta_{jkl}^r \end{aligned}$$

where $\Delta_{kl}^{qr_q}$ is a bound on the term $|P_w^{r_q}(A_{ik} = l) - P_w(A_{ik} = l)|$ for any user i. Notice that, as the MCVQ model suggests, the impact of a query rating on our predictions for a user i is solely mediated by its impact on the user's attitude vector.

A bound can be derived by assuming a "worst case" distribution over user attitudes, one that maximizes the impact of the query rating on the target product rating. We have

$$\begin{aligned} & |P_w^{r_q}(A_{ik} = l) - P_w(A_{ik} = l)| \\ & \quad = |\frac{(F + H_l) P_w(A_{ik} = l)}{\sum_{l'} (F + H_{l'}) P_w(A_{ik} = l')} - P_w(A_{ik} = l)| \\ & \quad = |\frac{(F + H_l)}{F + \sum_{l'} H_{l'} P_w(A_{ik} = l')} P_w(A_{ik} = l) - P_w(A_{ik} = l)| \end{aligned} \tag{3}$$

where F and $H_{l'}$ are defined as follows:

$$\begin{aligned} F &= \sum_{k' \neq k} P(T_q = k') \sum_{l' \le L} P_w(A_{ik'} = l') \theta_{qk'l'}^{r_q} \\ H_{l'} &= P(T_q = k) \theta_{qkl'}^{r_q} \end{aligned}$$

The user-dependent terms in this expression are the elements of the distributions $P(A_{ik'})$ for each k'. A "worst case user" requires us to set these distributions to maximize expression (3). We first note that F depends only the free variables $P(A_{ik'} = l'), k' \neq k$, and can be minimized independently of the distribution $P(A_{ik})$. Clearly minimizing F will maximize expression (3) for any setting of $P(A_{ik})$, and can be accomplished by setting $P_w(A_{ik'} = l_{k'}^*) = 1$ for $l_{k'}^* = \arg\min_{l'} \theta_{qk'l'}^{r_q}$.

We are left to set the distribution $P(A_{ik})$ to maximize expression (3). Let $H_{l_x} = \arg\max_{l'} H_{l'}$ and $H_{l_n} = \arg\min_{l'} H_{l'}$. The expression is maximized by assigning positive probability in $P(A_{ik})$ to only l and to either l_x—in which case the change in $P(A_{ik} = l)$ is maximally negative—or l_n—in which case the change in $P(A_{ik} = l)$ is maximally positive. Thus, this reduces to two distinct one-dimensional optimization problems (one for the maximum increase and one for the maximum decrease). The maximum decrease in $P(A_{ik}) = l$ can be found by maximizing the following expression w.r.t. $p = P(A_{ik} = l)$:

$$\frac{F + H_l}{F + pH_l + (1-p)H_{l_x}} p - p \tag{4}$$

Setting the derivative to zero, we obtain a positive solution at

$$p = \frac{H_l + F - \sqrt{FH_l + F^2 + FH_{l_x} + H_{l_x} H_l}}{H_l - H_{l_x}} \tag{5}$$

Analogous expressions exist for the maximum increase. We can thus set $\Delta_{kl}^{qr_q}$ to be the maximum (in absolute value) of the expressions for maximum increase and decrease.

The $\Delta_{kl}^{qr_q}$ can in turn be used to derive bounds on the influence of a query response on the mean rating of a target product j. Let $V_j^{qr_q}$ denote the posterior mean of R_{ij} given response r_q to query q, and V_j its prior mean. We can obtain a rough bound $\Delta_j^{qr_q}$ on $|V_j^{qr_q} - V_j|$ by noting that

$$|V_j^{qr_q} - V_j| \le \sum_r r \cdot \sum_k P(T_j = k) \sum_l \Delta_{kl}^{q r_q} \theta_{jkl}^r$$

This bound is too crude to be useful, since it assumes that all of the mass associated with different ratings r shifts in the same direction in response to the query.

We can derive a much tighter bound by explicitly modeling the prior of R_{ij} and finding a worst-case distribution $P(R_{ij})$ that maximizes $V_j^{qr_q} - V_j$.[7] This can be accomplished with a very compact linear program. We use variables $p_1, \ldots, p_\rho$ denoting the prior $P(R_{ij})$ of each rating $r \le \rho$; $q_1, \ldots, q_\rho$, denoting the posterior over R_{ij}; and δ_{kl} for each type-attitude pair k, l, denoting the actual change in $P(A_{ik} = l)$ in response to the query. We impose standard simplex contraints on the variables p_r and q_r. We also impose the bounds $-\Delta_{kl} \le \delta_{kl} \le \Delta_{kl}$. Finally, we relate the change in attitude distributions to the change in rating distributions by imposing the following equality constraint for each $r \le \rho$:

$$q_r - p_r = \sum_k P(T_j = k) \sum_l \theta_{kl}^r \delta_{kl}$$

Maximizing the objective function $\sum_r r \cdot (q_r - p_r)$ subject to these constraints bounds the change in mean rating. So we set $\Delta_j^{qr_q}$ to the objective value obtained by solving this LP.

[7] It isn't hard to show that the maximal increase in mean rating is identical in absolute terms to the maximal decrease, hence we concern ourselves only with the maximal increase.

The LP for each $\Delta_j^{qr_q}$ is very compact, with $KL + 2\rho$ variables, and $2KL + 5\rho + 2$ constraints. We do note that this bound can also be produced using a simple iterative algorithm with complexity $O(KL\rho)$ (we omit details). In practice, however, it appears that the direct LP formulation can be solved very effectively.

With this procedure in place, we can compute the set of terms Δ_j^{qr} for each product j, query (product) q, and query response (rating) r. While this computation is significant, again we emphasize that it is performed offline given a stable learned model, and is user-independent. These terms can be used to prune the number of posterior computations needed to compute the query with maximum EVOI. Let j^* be the product with highest mean rating for user i. For a specific query q, we can forego the computation of the posterior $P_w(R_{ij} = r|R_{iq} = r_q)$ (for each possible response r_q) if our bounds preclude the possibility of the mean of R_{ij} becoming higher than that of R_{ij^*}. More precisely, if we have

$$V_{j^*} - \sum_r P_w(R_{iq} = r)\Delta_{j^*}^{qr} \geq V_j + \sum_r P_w(R_{iq} = r)\Delta_j^{qr} \tag{6}$$

then we need not compute the posterior over R_{ij} when computing EVOI of q. As we will see, this can offer a significant degree of pruning.

We empirically evaluate the amount of pruning obtained by this approach using a procedure similar to the experiment presented in the previous section. We use the same trained MCVQ model as above. We observe w ratings of a given test user i, and update the attitude distributions and posterior over $\mathbf{r}^{unkn}(i)$, the ratings of unobserved items. For each possible query item q and target $j \in unkn(i)$, we compute Δ_j^{qr}, as well as $\Delta_{j^*}^{qr}$ for each q. For each movie j we can then apply Eq. 6 to determine if that movie cannot possibly obtain a higher rating than the model's current top-rated movie after query q. The number of movies satisfying this inequality describes the degree of pruning in posterior computations.

Figure 4 plots the pruning of potential targets as a proportion, calculated based on the ratio of number of unobserved movies not satisfying Eq. 6 to potential targets ($M - w - 1$). The figure shows a large degree of pruning at the early stages of (simulated) interaction with the user, but is fairly substantial throughout the interaction period. This implies that many items do not have the potential of ever surpassing the estimated utility of the model's top-ranked item, and substantial computational savings can be obtained by identifying these based on computations that can occur primarily offline. Again, while we show results only for users from the **main** subset, results from the **sparse** subset are qualitatively similar.

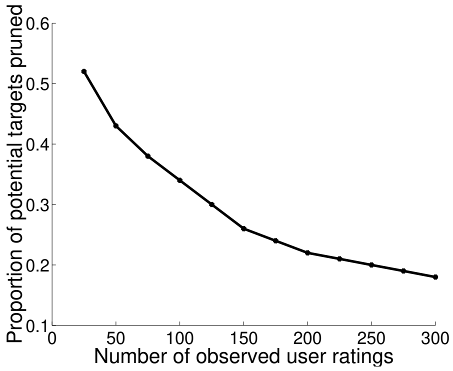

Figure 4: The proportion of unobserved items for which posterior distributions need not be computed is plotted for varying number of observed ratings of test users. As before, each datapoint an average of 5 test runs for each test user, with a random selection of observed ratings on each run.

5 Prototype Queries

The bounds in the previous section restrict the number of posterior computations over target products for each query-rating pair to those that could possibly become optimal; this reduces the $O(\rho M^2)$ problem to $O(\rho MN)$, where $N \leq M$ is the expected number of targets for which posteriors must be computed. This depends on the degree of pruning possible for a specific problem, but as we've seen, N appears to be considerably less than M in practice.

We might also attempt to reduce the number of queries we need to consider: considering $Q < M$ queries reduces online posterior computations to $O(\rho QN)$. In this section we describe a simple method for offline construction of a set of Q *prototype queries*, with the property that the EVOI of any query $m \in M$ is within some bound ε of some prototype query $q \in Q$. By restricting attention to queries in Q, we reduce online complexity further, but guarantee ε-optimal querying behavior.

Intuitively, the difference in the impact of two potential queries q and q' can be characterized by the difference in the type distributions of each query, and the difference in their rating parameters. For any product (i.e., potential query) q, define v_q to be a vector of length $KL\rho$ with elements $\Pr(T_q = k)\theta_{qkl}^r$. For two queries q and q', the fundamental distinction between q and q' can be characterized by the L_1-distance $d(q, q') = ||v_q - v_{q'}||_1$ between these vectors.

The key fact to notice is that the difference in Δ_{kl}^{qr} and $\Delta_{kl}^{q'r}$ (for any k, l, r) is bounded by a continuous function f of $d(q, q')$; that is, if $d(q, q') \leq \varepsilon$, then $|\Delta_{kl}^{qr} - \Delta_{kl}^{q'r}| \leq f(\varepsilon, k, l, r)$. We currently have some fairly crude bounds that are independent of all terms except ε, as well as a somewhat more reasonable approximation $f(\varepsilon, r) = 12\varepsilon/P(r)$, where $P(r)$ is the probability of receiving response r under query q.[8] From this, we can bound the difference between the terms Δ_j^{qr} and $\Delta_j^{q'r}$ for each target j with the same $f(\varepsilon)$. Finally, we obtain a bound on the difference in the expected mean rating change in target j due to query q and query q' via $\sum_r P(R_{iq} = r) f(\varepsilon, k, l, r)$. For example, we obtain

$$|V_j^q - V_j^{q'}| \leq 12\varepsilon$$

using $f(\varepsilon, r) = 12\varepsilon / P(r)$. Here V_j^q denotes the expected value (mean rating) of product j after receiving a response to query j.

This suggests an obvious method for constructing query protoypes that reduce the number of queries one needs to consider to guarantee that a query is chosen that has approximately optimal myopic EVOI. Given a learned model, our aim is to construct a set of prototype queries Q such that, for any product j, there exists a product $q \in Q$ such that $d(q, j) \leq \varepsilon$. This is a straightforward clustering task. If we restrict our attention to such a set and chose the query within Q that has maximum EVOI, we can guarantee that we are acting $f(\varepsilon)$-optimally with respect to considering the full set of potential queries.

We have not yet experimented with this approach, so we cannot comment on its efficacy. However, we expect that it can offer considerable savings. We emphasize again that the construction of a set of prototype queries can be done offline, allowing substantial online savings with respect to the number of required posterior computations.

6 Concluding Remarks

We have proposed an active approach to collaborative filtering, based on a probabilistic model of user preference data. Our framework is quite general, considering the value of queries that could most improve the quality of the recommendations made, based on the model's predictions. We have shown that offline pre-computation of bounds on value of information, and of prototypes in query space, can be used to dramatically reduce the required online computation. We also have derived detailed bounds for a particular model, and empirically demonstrated the value of our active approach using this model. While off-line computations should also lead to considerable savings in other probabilistic models, we expect the savings in MCVQ to be greater due to the user-independent assignment of movies to types.

Current directions of this work include improving the bounds, approximate pruning of targets, and further studies of prototyping of queries. In addition, we are examining costs models for queries, including modeling the probability that a user can answer a given query. Finally, we are considering extending the myopic approach to examine multistage lookahead, and offline policy construction.

[8] We expect that much tighter bounds than the ones we have derived currently are possible.

Acknowledgements

We thank Scott Helmer for his work on the movie database and on exploring other probabilistic methods of collaborative filtering. We also thank David Ross for his work on the multiple cause vector quantization algorithm. This research was supported by grants from NSERC and IRIS.

References

[Breese *et al.*, 1998] Jack S. Breese, David Heckerman, and Carl Kadie. Empirical analyis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 43–52, Madison, WI, 1998.

[Cohn *et al.*, 1996] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[Goldberg *et al.*, 2000] Ken Goldberg, Theresa Roeder, Dhruv Huptan, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. Technical Report M00/41, IEOR and EECS Departments, UC Berkeley, August 2000.

[Hofmann and Puzicha, 1999] Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 688–693, Stockholm, 1999.

[Konstan *et al.*, 1997] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: Applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[Nguyen and Haddawy, 1998] Hien Nguyen and Peter Haddawy. The decision-theoretic video advisor. In *AAAI-98 Workshop on Recommender Systems*, pages 77–80, Madison, WI, 1998.

[Pennock and Horvitz, 2000] David M. Pennock and Eric Horvitz. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 473–480, Stanford, 2000.

[Ross and Zemel, 2002] David Ross and Richard S. Zemel. Multiple cause vector quantization. In *Advances in Neural Information Processing Systems*, 2002. To appear.