Competitive Recommendation Systems

Petros Drineas [*]
Computer Science Department
Yale University

Iordanis Kerenidis [†]
Computer Science Division
University of California, Berkeley

Prabhakar Raghavan [‡]
Verity, Inc.
Sunnyvale, CA

ABSTRACT

A recommendation system tracks past purchases of a group of users to make product recommendations to individual members of the group. In this paper we present a notion of competitive recommendations systems, building on recent theoretical work on this subject. We reduce the problem of achieving competitiveness to a problem in matrix reconstruction. We then present a matrix reconstruction scheme that is competitive: it requires a small overhead in the number of users and products to be sampled, delivering in the process a net utility that closely approximates the best possible with full knowledge of all user-product preferences.

1. INTRODUCTION

Consider m users each of whom has a non-negative real *utility* for each of n products. This is conveniently viewed as an $m \times n$ matrix A underlying the preferences of users. These utilities (matrix entries) are not all known in advance; rather, we learn their values as users express preferences through transactions, marketing surveys, feedback ratings, etc. The goal of a recommendation system is to infer, based on the past actions of users, which products are likely to be of the greatest utility to each user. Such a system makes recommendations for a user based not only on his prior actions, but also on the actions of "similar" (in their past actions) users. In this latter guise, the process is sometimes known as *collaborative filtering*. There is a wealth of empirical literature [18, 21, 29, 30, 31, 32, 34, 35] on these ideas, and indeed quite a few commercial products have resulted.

The primary metric of performance here and in the prior literature is the quality (made precise below) of the recommendations. Computational performance metrics are also important, but typically less so than quality. Most algorithms used in practice are very computationally simple, and even those proposed in the theoretical literature entail only basic linear algebra. How does one evaluate these algorithms? Most empirical work [18, 21, 27, 30, 31] has employed the methodology of machine learning. In such evaluations one begins with a complete preference matrix. Some of its entries (usually chosen randomly) are deleted, and the remainder are fed to the recommendation algorithm as known preferences. The algorithm then recommends the products of highest utility to each user; its success is measured by some function that measures how much utility each user gets from the recommendations. Latent in this description is the notion that the algorithm is "reconstructing" the entries of highest utility in each row of the $m \times n$ preference matrix. In fact, many previous algorithms implicitly reconstruct approximations to *all* the entries of the matrix, even though entries of low value are not essential to the recommendation process.

1.1 Prior theoretical work

Kumar *et al.* [26] introduced a simple model in which each product fell into one of k equivalence classes. The preference matrix was then reduced to an $m \times k$ matrix, since all items within a class were assumed to have the same utility for a user. Under the further assumption that each user's utility vector (i.e., each row of the matrix) summed up to one, Kumar *et al.* [26] introduced a natural idea – that each row vector is sampled to yield that user's prior preferences. This captures the notion that prior data on a user is more likely to include products that have relatively high utility for the user. This paper gives simple algorithms for small values of k, measuring the sum of the utilities of the recommendations over the users. It also underscores that in order to provide good recommendations, an algorithm need not reconstruct all entries. It did however point out a number of unresolved issues: in practice products do not fall into equivalence classes within which all utilities are the same, the equivalence classes in the analysis had to be of roughly the same size, and the competitiveness of the algorithms rapidly diminished with increasing k.

Azar *et al.* [3] explicitly identified matrix reconstruction as central to recommendation systems, aiming to reconstruct the entire utility matrix (rather than only the highest values in each row). Provided the preference matrix had a rather strong "gap" property (see below), their procedure could reconstruct a good approximation to the entire matrix from a random sample of a constant fraction of the matrix. Their

[*]Part of the work was done while the author was a summer intern at Verity Inc. Also supported by NSF Grant CCR-9820850. E-mail: drineas@cs.yale.edu

[†]Part of the work was done while the author was a summer intern at Verity Inc. E-mail:jkeren@cs.berkeley.edu

[‡]E-mail: pragh@verity.com

work motivates the present work in two areas: understanding the gap requirement and avoiding sampling a constant fraction of the matrix.

Similar work has been done by Drineas *et. al.* [12, 13]. Although these papers assume full knowledge of the input matrices and then approximate Singular Value Decompositions or products of matrices, they can also be viewed as matrix reconstruction results. In particular, for a matrix A, [13] reconstructs the product AA^T given only a few columns of A. Observe that this result is not useful in our situation; we need to reconstruct A and not AA^T and there is no easy way to reconstruct A from AA^T. But, in [14] it is shown that A can be efficiently approximated using similar ideas from a sample of rows and columns of A. This algorithm (called CUR decomposition in [14]) is subsequently used for property testing in graphs. In Section 4, we adapt this algorithm to the problem of matrix reconstruction. Our goal is to show that we can efficiently approximate A using only partial information about A.

1.2 Our main results

In this paper we propose a notion of competitive recommendation systems, building on prior work [3, 26]. We then reduce the problem of competitive recommendation to a form of matrix reconstruction (intuitively motivated below, and formalized in Section 3). Next, we develop algorithms for the reconstruction problem that ask a "small" number of users for all their preferences, and the remaining users for preferences on a "small" number of products (Section 4). Here "small" (as we formalize below) is a number that is typically within a constant of the best possible on every instance, supporting our notion of competitive recommendation systems. Nonetheless, our algorithms are able to achieve a total utility close to the best possible with full knowledge of the matrix. As we will show, the number of samples of utilities is thus significantly smaller than in prior work. Moreover, some prior work [3, 28] requires a fairly stringent "gap" assumption on the preference matrix in order to apply Stewart's theorem [19] without motivating the nature of matrices generating such gaps. In avoiding this gap requirement we make recourse to a particular, plausible model for generating preference matrices. Finally, the running time of our algorithm is $O(mn)$; we briefly compare it with previous techniques in Section 4.

2. PRELIMINARIES

2.1 Competitive recommendations

As is clear from the preceding discussion, we have at least two important metrics for measuring the quality of an algorithm. First, as already mentioned, we must quantify the overall quality and utility of the recommendations. Second, we would like the algorithm to demand as little prior knowledge as possible, as measured by the number of samples of utility values. In analyzing the latter we adopt the view that although there are m users, the majority of them are likely to fall into a relatively small set of "types". Users of the same type need not have identical utility vectors, but do have relatively similar vectors – this is formalized below. Let k be the number of such types; we envision k as being considerably smaller than m. In practice we believe that k would typically be a constant independent of m; for the remainder of the paper we will refer to k as a constant in all qualitative discussion, even though our theorems will eventually be quantified in terms of their dependence on k.

We introduce the following notions in the spirit of competitive analysis. We will say an algorithm is *c-competitive for sampling* if it only uses prior information from ck rows and columns of the matrix. Intuitively (to within the constant c), one cannot hope for much better in this style of sampling — for otherwise, we risk the possibility of totally missing information on a type of user and thus failing to deliver any useful recommendations to that type. We say an algorithm is *f-competitive for utility* if it delivers a total utility (formally defined below) in recommendations that is at least f times the utility yielded by an algorithm that has full knowledge of the utility matrix. We thus seek algorithms that are *c-competitive for sampling* for a small value of c, and simultaneously *f-competitive for utility* for a large value of f. In fact, the algorithm we describe achieves these objectives even if *almost* all (rather than all) users fall into one of k dominant types.

2.2 Singular value decompositions

The *singular value decomposition* of an m by n matrix A is a factorization of the form:

$$A = U\Sigma V^T$$

where U and V are orthogonal $m \times m$ and $n \times n$ matrices and Σ is the diagonal matrix whose diagonal entries are the singular values σ_i of A. By $\sigma_i(A)$ we denote the i-th largest singular value of A. Let us define the $m \times n$ matrix A_k as

$$A_k = U_k\Sigma_k V_k^T = U_k U_k^T A = AV_k V_k^T,$$

where U_k is the $m \times k$ matrix consisting of the first k columns of U, Σ_k is the $k \times k$ diagonal matrix consisting of the k largest singular values, and V_k is the $n \times k$ matrix consisting of the first k columns of V.

Then, the matrix A_k is the best rank k approximation to A with respect to the 2-norm and the Frobenius norm of the matrix, i.e., the error $|A - A_k|_{2,F}$ is minimum over all possible k-rank approximations to A. See [19] for a more detailed discussion on SVD.

Also, the 2-norm of an $m \times n$ matrix A is defined as

$$|A|_2 = \max_{|x|_2=1} |Ax|_2$$

and the Frobenius norm as

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2}.$$

A well-known fact about the norms is that

$$|A|_2 \leq \|A\|_F.$$

Finally, for a matrix A, $A^{(i)}$ denotes the i-th column of A, $A_{(i)}$ the i-th row of A and $\mathrm{Tr}(A)$ the *trace* of A (the sum of its diagonal elements).

3. OUR MODEL AND SOME REDUCTIONS

We first define the notion of a "good" recommendation (Section 3.1). Next, we reduce the problem of good recommendations to a matrix reconstruction problem (Section 3.2). We then introduce the notion of important user types (Section 3.3). Finally, we show that if there exist relatively few

such types, making recommendations based on a low rank approximation of the preference matrix gives competitive results (Section 3.4). In what follows A is an $m \times n$ preference matrix and its rows are normalized (their length is 1).

3.1 Defining a "good" recommendation

Let $\hat{A}$ be the approximation derived by an algorithm for the matrix A. A recommendation for a user i consists of the top r elements of $\hat{A}_{(i)}$, where r is a reasonably small constant, say $5 \leq r \leq 10$. Denote by $a_{i1}, \ldots, a_{ir}$ the top r elements in the row $A_{(i)}$.

DEFINITION 1. *A product j is called* **good** *for user i if $A_{ij} > a_{ir} - \delta$, for some small constant δ.*

DEFINITION 2. *A recommendation for a user i is* **good** *if it contains at least one good product for the user.*

This is a somewhat weak definition of a good recommendation, but nevertheless a reasonable one. Intuitively a good recommendation is one that leads to a purchase. If we recommend five products to a user, one of which has high utility (i.e., very close to the top five products) then this is sufficient.

DEFINITION 3. *A recommendation algorithm is called f-competitive for utility, if it is* expected *to deliver* good *recommendations for at least $f \cdot m$ users.*

Obviously, $f \leq 1$. Also, a recommendation algorithm that has full knowledge of the preference matrix is 1-competitive. We will parameterize the competitiveness of our algorithm using the above definitions of a *good* recommendation and some properties of the preference matrix. Our goal is to show that, under rather natural assumptions, we can achieve high values of f while keeping the algorithm c-competitive for sampling for some small c. Under these definitions, the algorithm of [3] is $(1 - o(1))$-*competitive for utility* and $O(m/k)$-*competitive for sampling*. We should stress that the algorithm of [3] is $(1 - o(1))$-competitive for utility even under more stringent definitions of f-competitiveness. It is not clear though that such definitions make a recommendation system more useful; there should not be much difference between recommending the *highest* utility product and something close to it.

3.2 Good recommendations from good matrix reconstruction

Suppose that we have an approximation $\hat{A}$ that is close to A under the Frobenius norm. We will show that this implies guarantees on the quality of recommendations derived from $\hat{A}$. A bound in the Frobenius norm does not actually give element-wise guarantees for the approximation, but, for competitive recommendations, element-wise guarantees are not necessary. It suffices if the high valued elements of A remain high in the approximately reconstructed matrix $\hat{A}$. Note that the high valued elements of A cannot be distorted a lot, since they contribute to the error with respect to the Frobenius norm more than the light elements.

So let us assume that we have approximated the matrix A with $\hat{A}$ such that $\|A - \hat{A}\|_F^2 \leq \epsilon \|A\|_F^2$. We want an upper bound on the probability of a "bad" recommendation, which will give us the value of the f-competitiveness of our algorithm.

LEMMA 1. *Assuming that we have an approximation $\hat{A}$ to A such that $\|A - \hat{A}\|_F^2 \leq \epsilon \|A\|_F^2$, the probability of a bad recommendation is*

$$\Pr[\text{bad recommendation}] \leq \frac{2\epsilon}{r\delta^2}.$$

Proof: We note that δ and r come from Section 3.1. Then, a bad recommendation arises when all our recommendations for a user i, i.e., the r top elements of $\hat{A}_{(i)}$, have utility less than $a_{ir} - \delta$. We will compute the smallest contribution of a bad recommendation to the error $\|A - \hat{A}\|_F^2$. It is easy to see that the error takes the smallest value when the top r products in $A_{(i)}$ have utility u and the next r products all have utility $u - \delta$. In this case, we have to minimize the error over all values x, such that in $\hat{A}_{(i)}$ the r top products of $A_{(i)}$ have utility $\leq x$ and the r products with utility $u - \delta$ now have entries $\geq x$. This happens for $x = u - \frac{\delta}{2}$. The error in this case is

$$r \cdot (u - \frac{\delta}{2} - u)^2 + r \cdot (u - \delta - u + \frac{\delta}{2})^2 = \frac{r\delta^2}{2}.$$

Therefore, if we assume that λm users get a bad recommendation, the total error is $\lambda m r \delta^2/2$. Since we know that the error is bounded by $\epsilon \|A\|_F^2$ and $\|A\|_F^2 = m$ we get

$$\Pr[\text{bad recommendation}] \leq \frac{2\epsilon}{r\delta^2}.$$

□

In a plausible scenario, we can have an approximation with $\epsilon = 0.01$ and take $r = 10$ and $\delta = 0.1$. Then the above bound implies that the algorithm will give good recommendations for 80% of all users. Thus, our algorithm would be at least *0.8-competitive for utility*. Of course, the competitiveness of our algorithm also depends on the value of δ. In practice the algorithm will be very competitive even for worse values of the parameters (larger ϵ, smaller δ), since we anticipate the error to be distributed in a more random way amongst the elements and not adversarially.

This crude upper bound holds without making any assumptions about the utilities of the users. Making reasonable assumptions about these utilities, e.g., that there are a few high valued elements and many small elements in each row can give us better performance guarantees. Nonetheless, it is clear that a bound on the Frobenius norm of the "reconstruction" error will return competitive recommendations, since the high valued elements cannot be distorted a lot.

3.3 Modelling the users

In any large set of users, we would expect that most users belong to a small number of "well-separated" types. Then, a recommendation system will be competitive if it can produce good recommendations for this majority of users. We make this intuition more precise now in our model. We assume ℓ types of users (with ℓ possibly of the order of m), characterized by the vectors $v^{(1)}, \ldots, v^{(\ell)} \in [0,1]^n$. These types are unit vectors (their length is one) and "well-separated". Intuitively, this means that our types should not be linearly dependent or near-linearly dependent, since in this case some of the types are redundant.

DEFINITION 4. *The vectors $v^{(1)}, \ldots, v^{(\ell)}$ are δ-separated if for all pairs (i, j) such that $i \neq j$, ${v^{(i)}}^T \cdot v^{(j)} \leq \delta$.*

Observe that a set of types (each represented by a vector) that is 0-separated contains types that are orthogonal to each other and of unit length. If we define an $l \times n$ "matrix of types", then as types become more and more dependent on each other, the smallest singular value of this matrix shrinks. Thus, the matrix has a good approximation of rank less than ℓ or, equivalently, one of our types is close to being linearly dependent to the other types.

Imagine for the moment that all of our m users fall in one of the above types, so that our preference matrix has the following form:

$$A = \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_\ell \end{bmatrix}$$

where each of the T_i is a $t_i \times n$ matrix, with t_i identical rows (copies of $v^{(i)}$). Thus, t_i is the number of users that are of type i. Also, $\sum_{i=1}^{\ell} t_i = m$. Assume that the types are ordered so that $t_1 \geq t_2 \geq \ldots \geq t_\ell$. Although we have ℓ disjoint user types, we believe that most of the users belong to one of the few k "effective" user types. Intuitively, an effective user type is defined as a user type that contains a significant number of users.

DEFINITION 5. *A preference matrix in the form of A is called* (λ, k)-effective *if*

$$\sum_{i=1}^{k} t_i \geq \lambda \cdot m.$$

We will mostly be interested in matrices such that λ is close to 1 and $k = O(1)$. Thus, for such a matrix, most of its Frobenius norm ($\lambda \|A\|_F^2 = \lambda m$) is contained in the submatrices $T_1 \ldots T_k$. As a result, this matrix has an excellent low-rank approximation. More specifically,

LEMMA 2. *For a (λ, k)-effective matrix in the form of A,*

$$\|A - A_k\|_F^2 \leq (1-\lambda)\|A\|_F^2.$$

Proof: A_k is the optimal rank k approximation. By construction, using the above definitions, A has a rank k approximation (call it B_k) such that $\|A - B_k\|_F^2 \leq (1-\lambda)\|A\|_F^2$. More specifically, if B_k is the sum of k $m \times n$ matrices (each of rank 1)

$$B_k = \begin{bmatrix} T_1 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ T_2 \\ \vdots \\ \mathbf{0} \end{bmatrix} + \ldots + \begin{bmatrix} \mathbf{0} \\ \vdots \\ T_k \\ \vdots \end{bmatrix}$$

it follows that A_k can only be an even better approximation. □

Note that there may be up to $(1-\lambda)m$ users that belong to the remaining $\ell - k$ types. Assuming that our types are δ-separated, if the number of users that *do not* belong to an effective type is $\Omega(m)$ and δ is sufficiently small, then our matrix does not have an $\omega(\sqrt{m+n})$ gap as required in [3]. This is easy to see if our types are 0-separated. On the other hand [3] does not require our matrix form.

Following the lines of [28] and [3], we model the deviation of our users from their corresponding types by adding to A a matrix E. We define E_{ij} to be a random variable with mean zero and variance $\mathrm{Var}(E_{ij}) = O(\epsilon^2/m + n)$ for some small $0 < \epsilon < 1$. Thus, we define a new matrix

$$\tilde{A} = A + E.$$

This new matrix $\tilde{A}$ is our users-products matrix.

3.4 Bounds on the low rank approximation

Our first goal is to demonstrate that $\tilde{A}_k$ and A are close. Thus, if we could approximate the optimal low rank approximation to $\tilde{A}$, we could efficiently recreate the matrix A and thus accurately recommend a high utility product. Our analysis will allow us to bound the error of the approximation and closely follows similar analyses in [3, 28].

We now prove that if A is δ-separated, (λ, k)-effective and the top k types contain a significant number of users, then there is sufficient gap in A to apply Lemma 1 of [28].

LEMMA 3. *If $\sigma_1, \ldots \sigma_k$ are the singular values of A, A is δ-separated with $\delta = O(1/n)$ and $t_k/t_{k+1} \geq \beta_1 t_1/t_k$ for some large constant β_1, then*

$$\sigma_k/\sigma_{k+1} \geq \beta_2 \sigma_1/\sigma_k$$

for some constant $\beta_2 = O(\sqrt{b_1})$.

Proof: Our first step is to approximate the singular values of A. Observe that if A were 0-separated its singular values would be $\sqrt{t_1}, \ldots, \sqrt{t_\ell}$ and the above theorem would hold with $\beta_2 = \sqrt{\beta_1}$. The fact that A is not 0-separated slightly perturbs its singular values. We use the notation A_δ to denote a δ separated matrix A and A_0 to denote a 0 separated matrix A. Then, $A_\delta A_\delta^T$ is a matrix with entries at most δ when rows belonging to different types are multiplied and exactly 1 when rows belonging to the same type are multiplied. Thus, $A_\delta A_\delta^T - A_0 A_0^T$ has entries of absolute value at most δ. Thus, its 2-norm is at most $n \cdot \delta \leq O(1)$. Using standard perturbation theory results for the singular values of symmetric matrices (see e.g., [19]), it is easy to conclude that the singular values of A_δ are perturbed by at most $O(1)$. Choosing β_1 and β_2 carefully the result holds. We defer this to the final version of the paper. □

Now we can use Lemma 1 of [28] to prove that the rows $(\tilde{A}_k)_{(i)}$ and $(A_k)_{(i)}$ are close.

LEMMA 4. *Since $|E|_2 = O(\epsilon)$, then, with probability 1-$o(1)$, for all $i = 1 \ldots m$,*

$$|(\tilde{A}_k)_{(i)} - (A_k)_{(i)}|_2 \leq O(\epsilon)$$

Proof: Using Theorem 3 of [1], the 2-norm of E is at most $O(\epsilon)$ with probability 1-$o(1)$ (assuming that $m + n \geq 20$). Applying Lemma 1 of [28] or a similar lemma of [3] we see that $\tilde{V}_k = V_k R + G$ for some orthonormal matrix R and some matrix G such that $|G|_2 \leq O(\epsilon)$. Then, using $RR^T = I$,

$$\begin{aligned} & |(\tilde{A}_k)_{(i)} - (A_k)_{(i)}|_2 = \\ = \; & |\tilde{A}_{(i)} \tilde{V}_k \tilde{V}_k^T - A_{(i)} V_k V_k^T|_2 \\ = \; & |\tilde{A}_{(i)}(V_k R + G)(R^T V_k^T + G^T) - A_{(i)} V_k V_k^T|_2 \\ \leq \; & |(\tilde{A}_{(i)} - A_{(i)}) V_k V_k^T + \tilde{A}_{(i)} V_k R G^T|_2 \\ + \; & |\tilde{A}_{(i)} G R^T V_k^T + \tilde{A}_{(i)} G G^T|_2. \end{aligned}$$

But, since the 2-norm of any orthogonal matrix is 1, using the definition $\tilde{A}_{(i)} = A_{(i)} + E_{(i)}$ and the above derivation

$$\begin{aligned} & |(\tilde{A}_k)_{(i)} - (A_k)_{(i)}|_2 \leq \\ \leq\ & |\tilde{A}_{(i)} - A_{(i)}|_2 + 2O(\epsilon)|\tilde{A}_{(i)}|_2 + O(\epsilon^2)|\tilde{A}_{(i)}|_2 \\ \leq\ & |\tilde{A}_{(i)} - A_{(i)}|_2 + O(\epsilon)|\tilde{A}_{(i)}|_2 \\ \leq\ & |E_{(i)}|_2 + O(\epsilon)|E_{(i)}|_2 + O(\epsilon)|A_{(i)}|_2. \end{aligned}$$

Now, $|E_{(i)}|_2^2 = \sum_{j=1}^{n} E_{ij}^2$. Using our definitions, the expectation of $|E_{(i)}|^2$ is $O(n\epsilon^2/m+n) = O(\epsilon^2)$. A simple Chernoff bound shows that $|E_{(i)}|_2^2$ is tightly concentrated around its mean. Thus, with probability 1-$o(1)$,

$$|(\tilde{A}_k)_{(i)} - (A_k)_{(i)}|_2 \leq |E_{(i)}|_2 + O(\epsilon)|E_{(i)}|_2 + O(\epsilon) \leq O(\epsilon)$$

since $|A_{(i)}|_2 = 1$.

□

The following corollary is now obvious (simply sum the result of the above lemma for $i = 1 \ldots m$).

COROLLARY 1. *With probability* $1 - o(1)$, $\tilde{A}_k$ *and* A_k *are close w.r.t. the Frobenius norm, namely,*

$$\|\tilde{A}_k - A_k\|_F^2 \leq O(\epsilon)m = O(\epsilon)\|A\|_F^2.$$

Finally, if A is (λ, k)-*effective* it follows that $\|A - A_k\| \leq (1-\lambda)\|A\|_F^2$.

COROLLARY 2. $\tilde{A}_k$ *and* A *are close w.r.t. the Frobenius norm, namely,*

$$\|A - \tilde{A}_k\|_F^2 \leq (O(\epsilon) + 1 - \lambda)\|A\|_F^2.$$

3.5 Unknown $\tilde{A}$

In the previous section we showed that A and $\tilde{A}_k$ are indeed close. However we cannot compute $\tilde{A}_k$, since $\tilde{A}$ is not known! As proposed in [3] if the gap in A is $\omega(\sqrt{m+n})$, sampling $\Omega(mn)$ entries from $\tilde{A}$ suffice to approximate $\tilde{A}_k$ (and thus A) with $o(1)$ element-wise error. We seek an algorithm that samples $O(m+n)$ elements, within our user model. We briefly outline the algorithm and prove that it returns competitive recommendations when applied to the model of Section 3.3. We will prove the correctness of the algorithm using the users-products matrix A instead of $\tilde{A}$; the generalization of the proof to $\tilde{A}$ is straight forward.

In the following description k is the number of effective types, as defined in Section 3.3. To simplify the presentation, we assume that each effective type has a non-negligible number of users. More formally, we assume that $t_i \geq \lambda \cdot m/100k, i = 1 \ldots k$. Recall that t_i denotes the number of users with characteristic vector $v^{(i)}$.

1. Pick uniformly at random ak (out of m) users and ask for their utilities for all n products.

   We will see later that a is $O(\ln k)$.

2. Pick ak products and ask the remaining users their utilities about βk specific products.

   We will describe later how to pick these products; $\beta > 1$ is a constant.

3. Classify each user.

The above algorithm can be viewed as picking $O(k \log k(m+n))$ elements of the users-products matrix in order to approximate the missing utilities. These elements are not totally random elements; instead, we need a constant number of rows and columns. From a practical standpoint we believe that such sampling is in fact reasonable; we expand on this now.

A constant number of columns means that all the users who are interested in getting recommendations should initially answer a small questionnaire. Indeed, a user should expect to give out some information about his or her preferences, in order to acquire accurate and effective recommendations. Since we only need a constant number of columns, the size of the questionnaire will be small.

A constant number of rows means that we need to know the preferences of a few users on all products. Of course, we do not expect people to be willing to answer such questionnaires without compensation. However, it is common for companies to test the market by paying people to try out their new products. Note that we only need a small number of very typical users.

Let us assume that after running the above algorithm we identify the effective types $v^{(1)}, \ldots, v^{(k)}$ and let V be the $k \times n$ matrix of these vectors.

LEMMA 5. *If we randomly pick* $O(k \ln k)$ *users the probability of not observing all top* k *types is less that* 1%.

Proof: The probability of not observing all top k types is $\sum_{i=1}^{k}(1 - t_i/m)^{ak}$. Using $t_i \geq \lambda \cdot m/100k, i = 1 \ldots k$,

$$\begin{aligned} & \sum_{i=1}^{k}(1 - t_i/m)^{ak} = \\ \leq\ & \sum_{i=1}^{k}(1 - \lambda/100k)^{ak} \\ =\ & k(1 - \lambda/100k)^{(100k/\lambda)(a\lambda/100)} \\ \leq\ & ke^{-a\lambda/100}. \end{aligned}$$

It is easy to see that setting $a = 100\ln(100k)/\lambda$ reduces the probability of not observing all the effective types to 1%.

□

We will now describe steps 2 and 3 of the algorithm. We start by observing that given a user vector x we can classify the user into one of the k effective types based on the values of the dot products $x \cdot v^{(i)}$, where $v^{(i)}$, $i = 1 \ldots k$ are the k effective user types. Unfortunately, x is not fully known: instead we are only given βk elements of x. Recall that x is an n-vector of unit length. We now describe how to deal with this difficulty.

Once we have formed V (the $k \times n$ matrix of effective types), we pick a set of βk products (columns of V) to ask questions for, where the probability of picking a particular product $V^{(i)}$ is $|V^{(i)}|^2/\|V\|_F^2 = |V^{(i)}|^2/k, i = 1 \ldots n$. This essentially means that we ask questions about heavier products in V with higher probability. We decide on the above set of products once and we use it for all the remaining users. The following lemma shows that this amount of information suffices for the classification of a user.

LEMMA 6. *If we discover the utilities of* βk *products for a user-vector* x *(as described above), we can approximate* $V \cdot x$

by $\tilde{v}$ such that

$$|Vx - \tilde{v}|_2^2 \leq \frac{1}{\beta p}$$

holds with probability at least $1-p$, $0 < p < 1$.

Sketch of the proof: We only sketch a simple proof and defer a more elaborate proof — using martingales to get a factor of $\log p$ instead of p in the denominator — to the final paper.

We follow the lines of [13]. It suffices to analyze the following algorithm that approximates $V \cdot x$ by $\tilde{v}$, which is the product of $\tilde{V}$ (a $k \times \beta k$ matrix) and $\tilde{x}$ (a βk vector).

- **for $t = 1$ to βk independently**
  - Pick $i_t \in \{1 \ldots n\}$ at random with $p_i = \text{Prob}(i_t = i) = |V^{(i)}|^2/k, \quad i = 1 \ldots n$.
  - Include $V^{(i_t)}/\sqrt{\beta k p_{i_t}}$ as a column of $\tilde{V}$ and $x_{i_t}/\sqrt{\beta k p_{i_t}}$ as an element of $\tilde{x}$.
- Return $\tilde{v} = \tilde{V} \cdot \tilde{x}$ as the approximation to Vx.

Once a customer answers his questionnaire, we can compute $\tilde{x}$ and, since we know $\tilde{V}$ we can compute $\tilde{v}$. The rest of the proof follows from first principles: observe that $E(\tilde{v}) = V \cdot x$ and we can bound the variance of $\tilde{v}$ to conclude the proof.

□

COROLLARY 3. *If $\beta = 10/9p$ and our matrix is 0.1-separated, with probability at least $1-p$ we can exactly classify all the users belonging to the effective types. Thus, our algorithm is λ-competitive for utility.*

Unfortunately, although generalizing the above algorithm to arbitrary preference matrices (i.e., matrices not necessarily generated by the model of Section 3.3) is straightforward, a proof of its performance is not. The performance of this algorithm seems to depend a lot on the assumptions of our model and even small deviations from such assumptions cause substantial loss in competitiveness.

To sum up, our model assumes that most of the users belong to a small number of well-separated types. In this case, low-rank approximations allow for good recommendations for the users of the effective types. Therefore, it is reasonable to believe that a recommendation system that can accurately approximate the low-rank approximation of the matrix A with a small number of samples will be competitive. This is the subject of the next section.

4. APPROXIMATING A FROM SAMPLES

Based on the model described in Section 3.3 but also using the philosophical justifications of [3], we can assume that the user-products matrix A has a good low rank approximation. In this section, we will show that picking $O(k)$ rows and columns of the matrix is sufficient to learn the preferences of the users. Let us note that previous algorithms based on the SVD of the matrix A and standard perturbation results, need $\Omega(mn)$ elements of the matrix for a reconstruction with small error.

Our goal is to reconstruct the "hidden" $m \times n$ matrix A, based on partial information; namely only a constant number of rows and columns of A. We call the reconstructed matrix $\hat{A}$ and we provide error bounds for $\|A - \hat{A}\|_F^2$.

The intuition behind our algorithm is simple (see also [12, 13, 14]). If the Singular Value Decomposition of A is $A = U\Sigma V^T$, then the "optimal" rank k approximation is given by $A_k = U_k U_k^T A$, where U_k is the $m \times k$ matrix of the top k left singular vectors of A. In [12], we have essentially shown how to efficiently approximate A by

$$A \approx \tilde{U}_k \tilde{U}_k^T A \tag{1}$$

given only a few columns of A. The computation of $\tilde{U}_k$ does not involve A at all; only the sampled columns [1].

However, in the case of recommendation systems, A is "hidden". Thus, the algorithm of [12] is not sufficient. But, instead of using A in equation (1), we can use a matrix $\tilde{A}$ of the same dimensions such that $E(\tilde{A}) = A$. One could reasonably expect

$$A \approx \tilde{U}_k \tilde{U}_k^T A \approx \tilde{U}_k \tilde{U}_k^T \tilde{A} \tag{2}$$

We define $\tilde{A}$ as follows: its rows are either scaled versions of the corresponding rows of A (if these rows are revealed as part of the input) or all-zero vectors.

Before formally describing our algorithm, we should stress that we make *no assumptions* about the matrix A. Indeed, the algorithm (and its error bounds) apply to any matrix, regardless of its rank. Obviously, we will not assume that A conforms to the model of Section 3.3. The algorithm does not even need the user vectors (rows of A) to be of unit length; for clarity of exposition though we will retain this assumption.

We define two sets of probabilities: $p_i, i = 1 \ldots m$ and $q_j, j = 1 \ldots n$ such that:

1. $p_i \geq \alpha |A_{(i)}|^2 / \|A\|_F^2$, for some constant $\alpha \leq 1$.
2. $q_j \geq \beta |A^{(j)}|^2 / \|A\|_F^2$ for some constant $\beta \leq 1$.
3. $\sum_{i=1}^m p_i = \sum_{j=1}^n q_j = 1$.

Our algorithm assumes that we can sample rows (columns) of A with respect to the p_i (q_j). One could philosophically argue that such sampling is possible, even if A is unknown: our goal is to sample columns of A with probabilities proportional to their lengths, or, equivalently, products that seem more popular. This should be a rather realistic assumption, since we usually have some prior knowledge of a product's popularity over all users[2]. We emphasize that the exact lengths of rows and columns of A are not necessary; the use of α and β allows us to *approximate* them.

To make our exposition clearer, we assume that the rows of A are of unit length. Then, sampling rows of A (users) w.r.t. the p_i is trivial: uniform sampling will do ($\alpha = 1$). We now formally describe the algorithm.

Input:

1. r rows of A ($A_{(i_1)}, \ldots, A_{(i_r)}$), where the probability of picking the i-th row is p_i.
2. c columns of A ($A^{(j_1)}, \ldots, A^{(j_c)}$), where the probability of picking the j-th column is q_j.

Output: An $m \times n$ matrix $\hat{A}$ that approximates A.

[1] The goal of [12] was to efficiently approximate U_k, in less space and time than full SVD.

[2] A similar justification could be given for the rows of A.

The algorithm:

1. (*as in [12]*) We create an $m \times c$ matrix C with columns $A^{(j_t)}/\sqrt{cq_{j_t}}$, $t = 1 \ldots c$. We compute the top k left singular vectors of C and we denote them by the $m \times k$ matrix $\tilde{U}_k$.
2. We create an $m \times n$ matrix $\tilde{A}$ such that
$$\tilde{A}_{(i)} = \begin{cases} A_{(i)}/rp_i & \text{,if } i = i_t, t = 1 \ldots r \\ \mathbf{0^n} & \text{,otherwise} \end{cases}$$
where $\mathbf{0^n}$ is the all-zeros n-vector. Obviously, $E(\tilde{A}) = A$.
3. Return $\hat{A} = \tilde{U}_k \tilde{U}_k^T \tilde{A}$.

One could easily see that in forming $\tilde{A}$, all p_{i_t} are known and equal to $1/m$. Similarly, in forming C, $|A^{(j_t)}|$ is known (since this particular column is input to the algorithm). Thus q_{j_t} is also known, since $||A||_F^2 = m$.

Before stating our main theorem, we observe that $\tilde{A}$ can be alternatively defined as $\tilde{I} \cdot A$, where $\tilde{I}$ is an $m \times m$ diagonal matrix, such that

$$\tilde{I}_{ii} = \begin{cases} 1/rp_i & \text{, if } i = i_t, t = 1 \ldots r \\ 0 & \text{, otherwise.} \end{cases}$$

Our main theorem is:

THEOREM 1. *Let σ_t, $t = 1, \ldots, \rho$ denote the singular values of A (ρ is the rank of A). Then, using the above algorithm,*

$$E\left(\|A - \tilde{U}_k \tilde{U}_k^T \tilde{A}\|_F^2\right) \leq \sum_{t=k+1}^{\rho} \sigma_t^2 + \left(\sqrt{\frac{k}{\beta c}} + \frac{k}{\alpha r}\right) \|A\|_F^2.$$

Proof: Using $\tilde{A} = \tilde{I} \cdot A$ and adding and subtracting $\tilde{U}_k \tilde{U}_k^T A$ to the left-hand side,

$$\begin{aligned} & \|A - \tilde{U}_k \tilde{U}_k^T \tilde{I} A\|_F^2 = \\ = \; & \|A - \tilde{U}_k \tilde{U}_k^T A + \tilde{U}_k \tilde{U}_k^T A - \tilde{U}_k \tilde{U}_k^T \tilde{I} A\|_F^2 \\ \leq \; & \|A - \tilde{U}_k \tilde{U}_k^T A\|_F^2 + \|\tilde{U}_k \tilde{U}_k^T A - \tilde{U}_k \tilde{U}_k^T \tilde{I} A\|_F^2 \\ = \; & \|A - \tilde{U}_k \tilde{U}_k^T A\|_F^2 + \|\tilde{U}_k \tilde{U}_k^T (I - \tilde{I}) A\|_F^2. \end{aligned} \quad (3)$$

We now present two simple lemmas to bound the quantities on the right side of the above inequality.

LEMMA 7. *Given the notation of Theorem 1,*

$$E(\|A - \tilde{U}_k \tilde{U}_k^T A\|_F^2) \leq \sum_{t=k+1}^{\rho} \sigma_t^2 + \sqrt{\frac{k}{\beta c}} \|A\|_F^2.$$

Proof: This is essentially Lemma 4.1 from [12] (after an application of Markov's inequality to eliminate the expectation).

□

LEMMA 8. *Given the notation of Theorem 1,*

$$\|\tilde{U}_k \tilde{U}_k^T (I - \tilde{I}) A\|_F^2 = \|\tilde{U}_k^T (I - \tilde{I}) A\|_F^2.$$

Proof: Using that $\mathrm{Tr}(N^T N) = \|N\|_F^2$,

$$\begin{aligned} & \|\tilde{U}_k \tilde{U}_k^T (I - \tilde{I}) A\|_F^2 = \\ = \; & \mathrm{Tr}(A^T (I - \tilde{I}) \tilde{U}_k \tilde{U}_k^T \tilde{U}_k \tilde{U}_k^T (I - \tilde{I}) A) \\ = \; & \mathrm{Tr}(A^T (I - \tilde{I}) \tilde{U}_k \tilde{U}_k^T (I - \tilde{I}) A) \\ = \; & \|\tilde{U}_k^T (I - \tilde{I}) A\|_F^2. \end{aligned}$$

□

Before proceeding we observe that

$$(I - \tilde{I})_{ii} = \begin{cases} 1 - 1/rp_i & \text{, with probability } rp_i \\ 1 & \text{, with probability } 1 - rp_i. \end{cases}$$

Thus, $E\left[(I - \tilde{I})_{ii}\right] = 0$ and

$$E\left[(I - \tilde{I})_{ii}^2\right] = \frac{1}{rp_i} - 1 \leq \frac{\|A\|_F^2}{\alpha r |A^{(i)}|^2}.$$

We are now ready for

LEMMA 9. *Given the notation of Theorem 1,*

$$E(\|\tilde{U}_k^T (I - \tilde{I}) A\|_F^2) \leq \frac{k}{\alpha r} \|A\|_F^2$$

Proof:

$$\begin{aligned} & \|\tilde{U}_k^T (I - \tilde{I}) A\|_F^2 = \\ = \; & \sum_{i=1}^{k} \sum_{j=1}^{n} (\tilde{U}_k^T (I - \tilde{I}) A)_{ij}^2 \\ = \; & \sum_{i=1}^{k} \sum_{j=1}^{n} \left(\sum_{l=1}^{m} (\tilde{U}_k^T)_{il} (I - \tilde{I})_{ll} A_{lj} \right)^2 \\ = \; & \sum_{i=1}^{k} \sum_{j=1}^{n} \sum_{l_0=1}^{m} \sum_{l_1=1}^{m} (\tilde{U}_k^T)_{il_0} (I - \tilde{I})_{l_0 l_0} A_{l_0 j} (\tilde{U}_k^T)_{il_1} (I - \tilde{I})_{l_1 l_1} A_{l_1 j}. \end{aligned}$$

We observe now that $E\left[(I - \tilde{I})_{l_0 l_0} (I - \tilde{I})_{l_1 l_1}\right]$ is zero if $l_0 \neq l_1$. Also, the length of any row of $\tilde{U}_k^T$ is 1. Thus,

$$\begin{aligned} & E\left(\|\tilde{U}_k^T (I - \tilde{I}) A\|_F^2\right) = \\ = \; & E\left(\sum_{i=1}^{k} \sum_{j=1}^{n} \sum_{l=1}^{m} (\tilde{U}_k^T)_{il}^2 (I - \tilde{I})_{ll}^2 A_{lj}^2 \right) \\ = \; & \sum_{i=1}^{k} \sum_{l=1}^{m} (\tilde{U}_k^T)_{il}^2 E(I - \tilde{I})_{ll}^2 |A^{(l)}|^2 \\ \leq \; & \sum_{i=1}^{k} \sum_{l=1}^{m} (\tilde{U}_k^T)_{il}^2 \frac{\|A\|_F^2}{r\alpha |A^{(l)}|^2} |A^{(l)}|^2 \\ = \; & \sum_{i=1}^{k} \frac{\|A\|_F^2}{\alpha r} = \frac{k}{\alpha r} \|A\|_F^2. \end{aligned}$$

□

Combining Lemmas 9,7 and equation (3), we get the statement of Theorem 1. We could also present a bound for the 2-norm of the error, but it only adds intuition in our case. Such a bound is presented in [14], where it plays a central role.

□

Observe that Theorem 1 provides the guarantees that we are seeking; assuming that A has a good rank k approximation (e.g. $||A - A_k||_F^2 \leq \lambda \|A\|_F^2$),

$$E\left(\|A - \tilde{U}_k \tilde{U}_k^T \tilde{A}\|_F^2 / \|A\|_F^2\right) \leq \lambda + \left(\sqrt{\frac{k}{\beta c}} + \frac{k}{\alpha r}\right) = \lambda + \epsilon.$$

Thus, picking $O(k/\epsilon)$ rows and $O(k/\epsilon^2)$ columns bounds the expectation of the relative error of the approximation by $\lambda + \epsilon$. Using Markov's inequality, we see that picking $O(k/p\epsilon)$ rows and $O(k/p^2\epsilon^2)$ columns bounds the *relative error* of the approximation by $\lambda + \epsilon$ with probability $1 - p$ for any $p > 0$ [3].

Using the ideas of Section 3.2, suppose that we could fix $\epsilon + \lambda$ to 0.01 by picking $O(1)$ rows and columns. Then, using our definitions and considering $r = 10$ and $\delta = 0.1$, our algorithm is *0.8-competitive for utility* and $O(1)$*-competitive for sampling.*

Finally, the running time of the above algorithm is $O(mn)$. Indeed, we can compute the top k left singular vectors of C in $O(c^2m + c^3)$ time (by computing the left singular vectors of the $c \times c$ matrix C^TC). Then, in $O(mn)$ time we can compute the product $\tilde{U}_k\tilde{U}_k{}^T\tilde{A}$. The running time of the algorithm in [3] depends crucially on the accuracy and convergence of Lanczos methods, since they need to compute the SVD of a sparse $m \times n$ matrix and, obviously, full SVD is prohibitive.

5. CONCLUSIONS AND OPEN PROBLEMS

An important issue with the algorithm in Section 4 is the need to ask a small number of users about their preferences on all products. This, of course, is a tough task; in real life, few people might be willing to answer such a questionnaire.

We believe though that this particular difficulty is an interesting research problem. We expect to *compensate* users for their time to fill out such questionnaires. An interesting question is how to determine the appropriate payoff and, potentially, an answer could be given using game theoretic concepts. Similar work has been done in [25].

A different question is whether one could devise more efficient matrix reconstruction algorithms, that either achieve similar error bounds with less information or improve the error bounds of the above algorithms. We now state a result of [14], which essentially shows that the algorithm of Section 4 provides the best error bound one can hope for. For a detailed statement and proof of the following theorem we refer the reader to [14].

THEOREM 2. *Any algorithm returning a matrix $\hat{A}$ that approximates an $m \times n$ matrix A such that*

$$||A - \hat{A}||_{F,2} \leq \epsilon||A||_F$$

must output at least $O((n + m)\log(1/\epsilon))$ bits.

Thus, if we could reconstruct A by sampling less than $O(m + n)$ of its elements up to $\epsilon||A||_F$ error, then we could essentially "describe" A with less than $O(m+n)$ bits, an obvious contradiction to the above theorem. This does not directly imply the absence of algorithms returning more competitive recommendations; as we observed in our introduction, there might be algorithms returning competitive recommendations without reconstructing the entire matrix!

Acknowledgements: We wish to thank Ravi Kannan for many helpful discussions.

[3]We can strengthen the above result and replace p by $\log p$ using Martingale arguments; we defer this discussion to the final paper.

6. REFERENCES

[1] D. Achlioptas and F. McSherry. Fast Computation of Low Rank Approximations, *Proceedings of the 33rd Annual Symposium on Theory of Computing,* 2001.

[2] R.B. Allen. User models: Theory, method and practice. *International Journal of Man-Machine Studies*, 32:511–543, 1990.

[3] Y. Azar, A. Fiat, A. Karlin, F. McSherry and J. Saia Spectral analysis of data *Proc. of the 33rd ACM Symposium on Theory of Computing*, 2001.

[4] M.J. Berry and G. Linoff. *Data Mining Techniques.* John-Wiley, 1997.

[5] M.W.Berry, S.T.Dumais and G.W.O'Brien, Using linear algebra for intelligent information retrieval, *SIAM Review,* 37(4), 1995, pp. 573-595.

[6] J. Bettman. *An Information Processing Theory of Consumer Choice.* Addison-Wesley Publishing Company, 1979.

[7] R.C. Blattberg, R. Glazer, J.D.C. Little, eds. *The Marketing Information Revolution*, Harvard Business School Press, 1994.

[8] B. Bollobas. *Random Graphs.* Academic Press, NY, 1985.

[9] R. Boppana. Eigenvalues and graph bisection: An average-case analysis, *Proc. IEEE Symp. on Foundations of Computer Science*, 1987.

[10] M. Charikar, S.R. Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins. On targeting Markov segments. *Proc. ACM Symposium on Theory of Computing*, 1999.

[11] S. Deerwester, S. T. Dumais, T.K. Landauer, G.W. Furnas, and R.A. Harshman. Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41(6):391–407, 1990.

[12] P. Drineas, A. Frieze, R. Kannan, S. Vempala and V. Vinay, Clustering in large graphs and matrices, *Proceedings of the 10th Symposium on Discrete Algorithms*, pp. 291-299, 1999.

[13] P. Drineas and R. Kannan, Fast Monte-Carlo Algorithms for Approximate Matrix Multiplication, *Proceedings of the 42nd Annual Symposium on Foundations of Computing*, 2001.

[14] P. Drineas and R. Kannan, Pass Efficient Algorithms for Large Matrices, manuscript, 2001.

[15] A. Frieze, R. Kannan and S. Vempala, Fast Monte-Carlo algorithms for finding low rank approximations, *Proceedings of the 39th Annual Symposium on Foundations of Computing*, pp. 370-378, 1998.

[16] Z. Furedi and J. Komlos, The eigenvalues of random symmetric matrices, *Combinatorica* 1 (1981), pp. 233-241.

[17] R. Glazer. Marketing in an information-intensive environment: Strategic implications of knowledge as an asset, *Journal of Marketing*, 55:1–19, 1991.

[18] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM,* 35:12, pp. 51–60, 1992.

[19] G. Golub, C.F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 1989.

[20] W. Hoeffding, Probability inequalities for sums of bounded random variables, *American Statistical Association Journal*, March 1962, pp. 13-30.

[21] W. Hill, L. Stead, M. Rosenstein, G. Furnas. Recommending and evaluating choices in a virtual community of use. *Proceedings of ACM CHI,* pp. 194–201, 1995.

[22] D.L. Hoffman and T.P. Novak. Marketing in hypermedia computer-mediated environments: Conceptual foundations. *Journal of Marketing*, 60:50–68, 1996.

[23] J. Howard. *Consumer Behavior in Marketing Strategy*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[24] J. Kleinberg, C.H. Papadimitriou, P. Raghavan. Segmentation problems. *Proceedings of the ACM Symposium on Theory of Computing*, 1998.

[25] J. Kleinberg, C.H. Papadimitriou, P. Raghavan, On the Value of Private Information,*Proceedings of Theoretical Aspects of Reasoning about Knowledge*, 2001.

[26] R. Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins. Recommender systems: a probabilistic analysis. *Proc. 39th IEEE Symp. on Foundations of Computer Science*, 1998.

[27] B.N. Miller, J.T. Riedl, J.A. Konstan. Experiences with GroupLens: Making usenet useful again. *Proceedings of the USENIX Conference*, 1997.

[28] C.H. Papadimitriou, P. Raghavan, H. Tamaki and S. Vempala. Latent semantic indexing: A probabilistic analysis. *Proceedings of the ACM Symposium on Principles of Database Systems*, 1998.

[29] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews, *Center for Coordination Science, MIT Sloan School of Management Report WP #3666–94*, 1994.

[30] U. Shardanand. *Social Information Filtering for Music Recommendation*, Masters Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1994.

[31] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth", *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pp. 210–217, May 1995

[32] ACM SIGGROUP resource page on collaborative filtering. `http://www.acm.org/siggroup/collab.html.`

[33] L.G. Valiant. A theory of the learnable. *CACM* 27(11): 1134–1142, 1984.

[34] H.R. Varian. Resources on collaborative filtering. `http://www.sims.berkeley.edu/resources/collab/.`

[35] H.R. Varian and P. Resnick, eds. CACM Special issue on recommender systems. CACM 40(3), 1997.