A Theoretical Analysis of Query Selection for Collaborative Filtering

Sanjoy Dasgupta
Computer Science Department, University of California at San Diego (dasgupta@cs.ucsd.edu)

Wee Sun Lee
Computer Science Department, Singapore-MIT Alliance, National University of Singapore (leews@comp.nus.edu.sg)

Philip M. Long
Genome Institute of Singapore (gislongp@nus.edu.sg)

November 16, 2002

Abstract. We consider the problem of determining which of a set of experts has tastes most similar to a given user by asking the user questions about his likes and dislikes. We describe a simple algorithm for generating queries for a theoretical model of this problem. We show that the algorithm requires at most $\mathrm{opt}(F)(\ln(|F|/\mathrm{opt}(F)) + 1) + 1$ queries to find the correct expert, where $\mathrm{opt}(F)$ is the optimal worst-case bound on the number of queries for learning arbitrary elements of the set of experts F. The algorithm runs in time polynomial in $|F|$ and $|X|$ (where X is the domain) and we prove that no polynomial-time algorithm can have a significantly better bound on the number of queries unless all problems in NP have $n^{O(\log\log n)}$ time algorithms. We also study a more general case where the user ratings come from a finite set Y and there is an integer-valued loss function ℓ on Y that is used to measure the distance between the ratings. Assuming that the loss function is a metric and that there is an expert within a distance η from the user, we give a polynomial-time algorithm that is guaranteed to find such an expert after at most $2\mathrm{opt}(F,\eta)\ln\frac{|F|}{1+\deg(F,\eta)} + 2(\eta+1)(1+\deg(F,\eta))$ queries, where $\deg(F,\eta)$ is the largest number of experts in F that are within a distance 2η of any $f \in F$.

Keywords: collaborative filtering, recommender systems, membership queries, approximation algorithms,inapproximability

1. Introduction

Recommender systems (also known as collaborative filtering systems) use the opinions of past users to make recommendations to new users. The design of many such systems is based on the assumption that people with similar opinions about some things are likely to have similar opinions about others (see (Resnick and Varian, 1997; Breese et al., 1998)). The user is typically asked to rate a few items before any new item is recommended. Once a sufficient number of items have been rated, the system can use those ratings to estimate which previous users

of the system are most similar to the current user overall. The opinions of these previous users can then be used to generate recommendations; methods based on weighted majority prediction (Nakamura and Abe, 1998) and correlation coefficients (Resnick et al., 1994) usually work quite well for this.

In this paper, we investigate a different aspect of the problem: how to select the initial items for the user to rate. These items are not presented as recommendations, but are asked only for the purpose of learning about the user. Since these are troublesome to the user, a high priority must be placed on asking few of these questions. (This is in contrast to the work on "approximate nearest neighbor searching" (Arya et al., 1994; Kushilevitz et al., 1998; Indyk and Motwani, 1998), where all the components of the point being searched are assumed to be given.) If later questions are decided based on the answers to earlier questions, the questions must also be generated in real time. To our knowledge, this problem has not been studied in the collaborative filtering literature before.

We allow the ratings to come from any finite set Y, and assume that the algorithm is given an integer-valued loss function ℓ on Y to measure the distance between different ratings. We require that the loss function ℓ be a metric, that is, it satisfies the properties: $\ell(x, y) \geq 0$, $\ell(x, y) = 0$ if and only if $x = y$, $\ell(x, y) = \ell(y, x)$ and $\ell(x, y) \leq \ell(x, z) + \ell(z, y)$ for any $z \in Y$. Common loss functions that satisfy these properties include the $0 - 1$ loss and the absolute loss. The distance between users will then be measured by the sum, over all items, of the loss between their ratings on a given item.

To emphasize the role that they play, we refer to the previous users as *experts*; our approximation bounds will be in terms of the number of such experts. Our analyses assume that each of these experts have rated all items in the system. In practice, some users can be paid to rate all the items. Alternatively, the system may be able to generate such experts by clustering previous users, each of whom may have rated only some of the items, and using the cluster centers as the experts.

Our algorithm and its analysis in the general case are easiest to understand as extensions of a simple algorithm and analysis for an artificial, highly idealized setting. In this setting, we assume that there are only two possible ratings, that the distance between ratings is 1 if they are different and 0 if they are the same, and that some expert agrees with the user on all items. In this case, the problem can be described in terms of the *membership query model* (Angluin, 1988).

In the membership query model (Angluin, 1988), the learning algorithm is trying to learn an unknown $\{0, 1\}$-valued function f (called the "target") chosen from a known concept class F. The algorithm is

allowed to ask the value of $f(x)$ for domain elements x of its choosing, and must eventually halt and output the identity of f. In the idealized case described above, the problem of finding the perfect expert can be viewed as the problem of learning using membership queries. The different items would be the domain X, the likes and dislikes of the user is the function f to be learned, and asking the user its opinion about an item can be interpreted as a membership query. The experts are then the concept class F. Viewed this way, the problem we are faced with is that of, given a concept class F as input, designing a membership query algorithm for F.

The number of queries required for learning in the membership query model has been studied in (Bshouty et al., 1996; Hellerstein et al., 1996; Hegedus, 1995; Gortler and Servedio, 2001). These studies typically try to identify parameters of function classes that characterize the number of queries required. They also give bounds on the number of queries required for various function classes, typically boolean circuits. In this work, our function classes are meant to model a set of experts. Hence we assume only that we have a finite class F and try to derive algorithms that perform well relative to $\mathrm{opt}(F)$, the optimal worst-case bound on the number of membership queries for learning arbitrary elements of F. This particular problem was studied in (Arkin et al., 1998) (see (Angluin, 2001)). They showed that the very simple and fast "query-by-committee" (Seung et al., 1992) algorithm, which maintains a list of possible targets, and chooses the query for which the remaining possibilities are most evenly divided, learns any class F with an approximately optimal number of membership queries in the worst case. Specifically, they proved that the query-by-committee algorithm learns arbitrary elements of F while making at most $\mathrm{opt}(F)\log_2|F|$ queries.

In this paper, we provide a simpler proof of an improved bound of $\mathrm{opt}(F)(\ln(|F|/\mathrm{opt}(F))+1)+1$ queries. While this is a modest improvement for the basic problem, (the leading constant is improved by a factor of $1/(\ln 2)\approx 1.44$), the structure of the proof forms the basis for our analysis of the general case.

A result in (Hyafil and Rivest, 1976) (see (Angluin, 2001)) directly implies that, if $P\neq NP$, there is no polynomial-time algorithm that learns arbitrary elements of F with an exactly optimal number of membership queries in the worst case. We show that, if not all problems in NP have $n^{O(\log\log n)}$ time algorithms, it is impossible to design a polynomial-time algorithm that, given F as input and a membership oracle for an element f of F, is guaranteed to learn f using $\mathrm{opt}(F)((1/2-o(1))\ln|F|)$ queries.

Next, we look at the more general case. To study this case, we use a variant of the membership query model similar to that proposed in (Angluin et al., 1997). Here, the range of the target f (our model of the user) and the functions in F (the experts) is an arbitrary finite set Y. As mentioned above, the algorithm is given an integer-valued metric ℓ on $Y \times Y$, and the distance between functions f and g is measured by $\sum_{x \in X} \ell(f(x), g(x))$. The target function f is not necessarily in F, but the algorithm is given a parameter η such that there is a function g in F at a distance at most η from f. The algorithm must output some element of F within distance η (there may be more than one). Let us refer to the optimal worst-case bound on the number of queries for this model by $\mathrm{opt}(F, \eta)$.

The algorithm we analyze for this problem also maintains a list of elements of F that are "alive"; here, these are elements that might possibly be within distance η of the target function f. Loosely speaking, it repeatedly chooses a domain element for which any response will discredit the remaining possibilities in total by a large amount.

To analyze this algorithm, we make use of a quantity that we call the η-degree of F. In the recommender system application, this can be interpreted as a measure of the diversity of opinion among the experts; for example, if any possible target f is at a distance at most η from a unique element of F, then the η-degree of F is 0. The motivation for this measure is strongest if we imagine that F is the result of a clustering preprocessing step. Note that, informally, if F consists of the centers of tight clusters, and users typically belong to one such cluster, being much closer to one element of F than to any other should often be expected in practice. Tight clustering is the implicit assumption underlying the design of many collaborative filtering systems.

One can view the definition of η-degree as follows: imagine centering balls of radius η at the elements of F, and constructing a graph where the vertices are these balls, and there are edges between pairs of vertices that overlap. The η-degree of F, denoted $\deg(F, \eta)$, is the edge degree of that graph.

Our generalization of the query-by-committee algorithm is guaranteed to find an element of F within distance η after at most

$$2\mathrm{opt}(F, \eta) \ln \frac{|F|}{1 + \deg(F, \eta)} + 2(\eta + 1)(1 + \deg(F, \eta))$$

queries. Thus, if each possible target is within distance η of a unique element of F,

$$2\mathrm{opt}(F, \eta) \ln |F| + 2(\eta + 1)$$

queries suffice. The algorithm runs in time polynomial in $|X|$, $|Y|$ and $|F|$. Since the idealized case is a special case with $\eta = 0$ and $Y = \{0, 1\}$,

finding a polynomial-time algorithm with significantly better bound on the number of queries does not appear to be possible when $\deg(F, \eta)$ is small.

2. Membership Queries

Fix some finite domain X. For some function f from X to $\{0, 1\}$, a membership oracle for f, when queried about an element x of X, returns $f(x)$. For an algorithm A with access to a membership oracle for f, let $Q(A, f)$ be the number of queries asked by A before it outputs the identity of f. For a class F of functions from X to $\{0, 1\}$, let $Q(A, F)$ be the maximum of $Q(A, f)$ over all $f \in F$. Let $\mathrm{opt}(F)$ be the minimum of $Q(A, F)$ over all algorithms A (note that there is no limitation on the time taken by A).

In this section, we show that there is an algorithm that takes F as input, and, given access to an oracle for an arbitrary element f of F, learns f with a nearly optimal number of queries in time polynomial in $|F|$ and $|X|$.

2.1. Analysis

The algorithm analyzed in this section is the "query-by-committee" (Seung et al., 1992) algorithm. (Our analysis of it builds on Johnson's analysis of his approximation algorithm for Set Cover (Johnson, 1974).) It maintains a list of the elements of F consistent with the answers received so far, and asks the query x that divides the elements the most evenly, i.e. for which the number of "alive" functions g for which $g(x) = 1$ and the number for which $g(x) = 0$ are as close as possible. After receiving $f(x)$, those possibilities that are inconsistent with this value are deleted, and the algorithm continues. When only one possibility remains, the algorithm halts and outputs it.

The key lemma in our analysis is the following.

LEMMA 1. *For any domain X, and any finite set F of at least two functions from X to $\{0, 1\}$, there is an x for which* $\min_{y \in \{0,1\}} |\{f \in F : f(x) = y\}| \geq (|F| - 1)/\mathrm{opt}(F)$.

Proof: Let A be an optimal membership query algorithm for F. Assume for the sake of contradiction that for all $x \in X$, either $|\{f \in F : f(x) = 1\}| < (|F| - 1)/\mathrm{opt}(F)$, or $|\{f \in F : f(x) = 0\}| < (|F| - 1)/\mathrm{opt}(F)$.

Our strategy will be to use the fact that any possible query that A could ask has an answer that eliminates few possibilities to argue that

after asking a certain number of queries, A cannot know the function to be learned. We will design an adversary that repeatedly gives the answer to A that eliminates the fewest possibilities.

Let $F_0 = F$ (in general, F_t will be the possibilities remaining after t queries have been asked). Let x_1 be the first query asked by A. Choose y_1 to maximize $|\{f \in F : f(x_1) = y_1\}|$. Let $F_1 = \{f \in F : f(x_1) = y_1\}$. Then, by assumption, $|F_1| - 1 > |F| - 1 - (|F| - 1)/\text{opt}(F)$.

Continuing, let each x_t be the t^{th} query asked, and choose y_t to maximize $|\{f \in F : f(x_t) = y_t\}|$, and let $F_t = \{f \in F : f(x_1) = y_1, ..., f(x_t) = y_t\}$. For each such t, $|F_t| - 1 > |F_{t-1}| - 1 - (|F| - 1)/\text{opt}(F)$. Telescoping,

$$|F_{\text{opt}(F)}| - 1 > (1 - \text{opt}(F)/\text{opt}(F))(|F| - 1) = 0. \tag{1}$$

Thus, after $\text{opt}(F)$ queries, there is more than one element of F consistent with the information received by A, a contradiction. □

THEOREM 2. *For any finite set X, and any finite set F of at least two functions from X to $\{0,1\}$, the query-by-committee algorithm, given F and a membership oracle for any arbitrary $f \in F$, outputs f after asking at most*

$$\text{opt}(F)\left(1 + \ln\frac{|F| - 1}{\text{opt}(F)}\right) + 1$$

queries.

Proof: Choose F, and a target $f \in F$. Suppose the query-by-committee algorithm asks T queries before learning f, and for each $0 \leq t \leq T$, let F_t be the set of functions in F consistent with the information received after the first t queries. Lemma 1 implies that for all $t \leq T$,

$$\begin{aligned} |F_t| - 1 &\leq (1 - 1/\text{opt}(F_{t-1}))(|F_{t-1}| - 1) \\ &\leq (1 - 1/\text{opt}(F))(|F_{t-1}| - 1). \end{aligned} \tag{2}$$

We also have

$$|F_t| \leq |F_{t-1}| - 1. \tag{3}$$

Let S be the largest index for which $|F_S| - 1 \geq \text{opt}(F)$. Then (2) implies

$$\left(1 - \frac{1}{\text{opt}(F)}\right)^S (|F| - 1) \geq \text{opt}(F)$$

and, applying the fact that $\forall x, 1 - x \leq e^{-x}$ and solving for S, we get

$$S \leq \text{opt}(F) \ln\frac{|F| - 1}{\text{opt}(F)}.$$

Also, (3), together with the fact that $|F_{T-1}| > 1$, implies that

$$T - S < \text{opt}(F) + 1,$$

completing the proof. □

2.2. Hardness of approximation

An approximation algorithm for a minimization problem achieves an approximation ratio of $\alpha > 1$ if for every instance of the problem, it computes a solution that costs no more than $\alpha \cdot OPT$, where OPT is the optimum (minimum) cost for the problem instance. We will show that there is no polynomial-time algorithm with an approximation ratio of $(\frac{1}{2} - o(1)) \ln |F|$ for finding the minimum number of queries unless all problems in NP have $n^{O(\log \log n)}$ time algorithms.

The result will be proven by using an approximation-preserving reduction from the Set Cover problem.

Set Cover

Input: Base set B with $|B| = n$; subsets $\mathcal{S} = \{S_1, \ldots, S_m\}$ of B.

Output: $\mathcal{T} \subset \mathcal{S}$ which covers B, i.e. for which $B \subseteq \bigcup_{T \in \mathcal{T}} T$.

Goal: Minimize $|\mathcal{T}|$.

LEMMA 3 (Feige, 1998 (Feige, 1998)). *If, for some $\epsilon > 0$, there is a polynomial-time algorithm which approximates* Set Cover *within $(1 - \epsilon) \ln n$, then $NP \subset TIME(n^{O(\log \log n)})$.*

THEOREM 4. *Unless $NP \subset$ TIME$(n^{O(\log \log n)})$, there is no polynomial-time algorithm which guarantees a $(\frac{1}{2} - o(1)) \ln |F|$-approximation to the problem of learning arbitrary elements of F with a minimal number of queries.*

Proof: Given an instance of Set Cover, let $n = |B|$ and let $x_1, \ldots, x_n$ denote the elements of B. Assume without loss of generality that no two elements of B are present in exactly the same subsets S_j (redundant elements can be discarded). Create an $n \times m$ matrix A whose $(i, j)^{th}$ entry is

$$A_{ij} = \begin{cases} 1 & \text{if } x_i \in S_j \\ 0 & \text{otherwise} \end{cases}$$

Now consider the large $(n^2 + n + 1) \times (mn + n)$ matrix in Figure 1. The unshaded portions are all zero.

We will construct an instance of our query problem in which each column of the matrix represents a possible query, each row represents

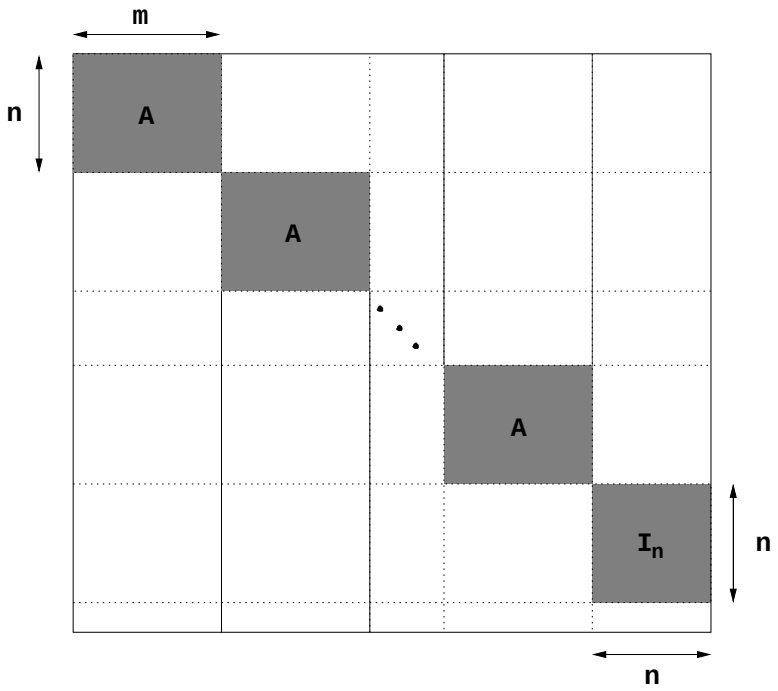

Figure 1. Reduction from SET COVER to our problem. There are n copies of matrix A. All unshaded portions of this large $(n^2 + n + 1) \times (nm + n)$ matrix are zero. I_n denotes the $n \times n$ identity matrix.

a function $f \in F$, and the entries of the matrix actually specify all the functions.

Suppose the original instance has a minimum set cover of size s. Now suppose some $f \in F$ is chosen and the goal is to identify it.

1. Consider any of the first n blocks of n functions (each block corresponds to a copy of matrix A in Figure 1). If f has a nonzero value in one of the columns of this block, let us say that f lies in the block. We can determine whether f lies in a given block by asking queries corresponding to columns that form a cover of the rows in the block – f lies in the block if and only if any of the answers to those queries is 1. Furthermore, to eliminate all the functions in a block, questions corresponding to columns that cover the rows must be asked – otherwise, some function f in the block would be consistent with all 0 answers. Thus, in order to determine whether f lies in this block, at most s queries are needed, and in order to eliminate all the functions in a block from consideration, exactly s queries are needed.

2. In order to eliminate the last block of n functions (corresponding to matrix I_n in Figure 1), exactly n queries are needed.

3. Therefore, if the target function f is the last row, then $ns + n$ queries are needed.

4. If the target function is any other row, at most this many queries are needed. Basically, ns queries are sufficient to identify the correct block of functions, and a further n queries will pinpoint the correct function.

Therefore, the optimal number of queries is exactly $ns+n$. Suppose we have an α-approximation algorithm for our problem; give it this instance, and ask it to identify the last function (the last row). At least n of its queries must be from the last block (corresponding to I_n), and so there must be at least one of the first n blocks which receives $\leq (\alpha(ns+n)-n)/n = \alpha(s+1)-1$ queries. These $\alpha(s+1)-1$ queries yield a set cover of the original instance (otherwise that particular block of functions cannot be completely eliminated). It follows from Lemma 3 that $\alpha > (1-o(1))\ln n = (1/2-o(1))\ln|F|$, unless NP has slightly superpolynomial-time algorithms. □

3. The General Case

Choose a finite nonempty set Y, and a metric ℓ mapping $Y \times Y$ to the nonnegative integers.

For functions f and g from X to Y define the distance $d(f,g)$ between f and g by

$$d(f,g) = \sum_{x \in X} \ell(f(x), g(x)).$$

Let $\mathcal{B}(F,\eta)$ consist of all $g : X \to Y$ such that there is an $f \in F$ for which $d(f,g) \leq \eta$.

In this model, the adversary picks a function g from $\mathcal{B}(F,\eta)$ (which we will call the target), and provides an evaluation oracle for g to the algorithm; this oracle responds to a query of x with $g(x)$. The algorithm then must output $h \in F$ such that $d(g,h) \leq \eta$ (there may be more than one such possibility). The worst case number of queries for an algorithm A in this setting is $Q(A,F,\eta)$, and the optimal number of queries is $\mathrm{opt}(F,\eta)$.

For a function $f : X \to Y$, and $U \subseteq X$, denote the restriction of f to U by $f_{|U}$. For a set F of functions from X to Y, define the η-degree

of F, denoted by $\deg(F, \eta)$, to be the maximum, over all $h \in F$, of $|\{f \in F : 0 < d(f, h) \leq 2\eta\}|$.

For technical reasons, we will consider a related model. In this model, instead of η, the learning algorithm is given a priori a function $\mu : F \to \mathbf{Z}^+$ called a *quota function*, and access to an evaluation oracle for some $g : X \to Y$ such that there is a $h \in F$ with $d(g, h) \leq \mu(h)$. The quota function is used to indicate how much error the function is allowed to make before being eliminated.

The algorithm must output an $h \in F$ such that $d(g, h) \leq \mu(h)$. Let $\mathrm{opt}(F, \mu)$ be the optimal worst-case number of queries for learning in this model. For $\mu : F \to \mathbf{Z}^+$, define $\phi(F, \mu)$ to be the maximum, over all $h \in F$, of

$$\sum_{f \in F : d(f,h) \leq \mu(f) + \mu(h)} (1 + \mu(f)).$$

The value $\phi(F, \mu)$ bounds the amount of quota of functions near the target. These functions are harder to eliminate and are treated separately in the proofs.

3.1. Algorithm

Our algorithm works in time polynomial in $|X|$, $|Y|$, and $|F|$. (Note that the latter is significantly less than $|\mathcal{B}(F, \eta)|$.)

Let x_i and y_i, $i < t$, be the queries and responses before time t and let

$$F_t = \left\{ f_{|_{X - \{x_1, \ldots, x_{t-1}\}}} : f \in F, \sum_{s<t} \ell(f(x_s), y_s) \leq \mu(f) \right\}.$$

Informally F_t consists of the restrictions of those elements of F that are "still alive" to the unexplored portion of X. For each t, define $\mu_t : F_t \to \mathbf{Z}^+$ by

$$\mu_t(h) = \max \left\{ \mu(f) - \sum_{s<t} \ell(f(x_s), y_s) : f \in F, f_{|_{X - \{x_1, \ldots, x_{t-1}\}}} = h \right\}.$$

Informally, $\mu_t(h)$ is the amount of loss left before h is eliminated as a possible target.

For $f \in F$, define $\ell_{f,t} : X \times Y \to \mathbf{Z}^+$ by

$$\ell_{f,t}(x, y) = \min \left\{ \ell(f(x), y), (\mu(f) + 1) - \sum_{s<t} \ell_{f,s}(x_s, y_s) \right\}.$$

We can think of $\ell_{f,t}$ as a "truncated" version of the loss function ℓ, in which if the loss at trial t is enough to put f over its "loss budget", the loss is reduced to the smallest amount that does this.

For $f \in F_t$, define $\ell'_{f,t} : X - \{x_1, \ldots, x_{t-1}\} \times Y \to \mathbf{Z}^+$ by

$$\ell'_{f,t}(x, y) = \min\{\ell(f(x), y), \mu_t(f) + 1\}.$$

The function $\ell'_{f,t}$ is like $\ell_{f,t}$, except that it concerns F_t and μ_t. Each function $f \in F_t$ has a corresponding function $\bar{f} \in F$ (which is an extension of f to $\{x_1, \ldots, x_{t-1}\}$) such that $\ell'_{f,t} = \ell_{\bar{f},t}$ on $X - \{x_1, \ldots, x_{t-1}\}$.

Our algorithm (let's call it $B_{F,\mu}$, and, in the case that $\mu(f) = \eta$ for all f, $B_{F,\eta}$), is defined as follows. In round t, if $|F_t| = 1$, the algorithm halts, and outputs an extension h of the single element of F_t to all of X that minimizes $\sum_{s<t} \ell(h(x_s), y_s)$. Otherwise, it chooses x_t from $X - \{x_1, ..., x_{t-1}\}$ in order to maximize

$$\min_{y \in Y} \sum_{f \in F_t} \ell'_{f,t}(x_t, y).$$

The algorithm is similar to the algorithm in Section 2 in that it picks the element that will maximize the smallest (over the labels) sum of losses of the remaining functions. The main difference is that functions are eliminated when they exceed their loss quota or when they are identical to other functions in F_{t-1} on $X - \{x_1, ..., x_{t-1}\}$ but have larger cumulative losses.

3.2. Analysis

The following is the main lemma in our analysis of this algorithm. As in Lemma 1, it shows that it is always possible to select an element in the set that will induce a large sum of losses (over the functions, hence eliminating the functions quickly) regardless of the label associated with the element.

LEMMA 5. *Choose a finite X, a finite Y, a finite set F of at least 2 functions from X to Y, and $\mu : F \to \mathbf{Z}^+$. There is an $x \in X$ for which*

$$\min_{y \in Y} \sum_{f \in F} \ell'_{f,1}(x, y) \geq \frac{1}{\mathrm{opt}(F, \mu)} \left(\left(\sum_{f \in F} (1 + \mu(f)) \right) - \phi(F, \mu) \right)$$

Proof: Let A be an optimal membership query algorithm for learning F with a quota function μ in the model of this section. Let $T = \mathrm{opt}(F, \mu)$. Assume without loss of generality that A always asks exactly T queries before halting and outputting a function in F. Assume for the sake of contradiction that

$$\forall x, \exists y, \sum_{f \in F} \ell'_{f,1}(x, y) < \frac{1}{T} \left(\left(\sum_{f \in F} (1 + \mu(f)) \right) - \phi(F, \mu) \right). \quad (4)$$

Note that

$$\sum_{f \in F} \ell'_{f,1}(x,y) = \sum_{f \in F} \ell_{f,1}(x,y)$$

and that, for all $t \geq 1$,

$$\sum_{f \in F} \ell_{f,t}(x,y) \leq \sum_{f \in F} \ell_{f,1}(x,y). \tag{5}$$

Generate $(x_1, y_1), ..., (x_T, y_T)$ recursively as follows. For each t, let x_t be A's query when its previous queries $x_1, ..., x_{t-1}$ were answered with $y_1, ..., y_{t-1}$ respectively. Choose

$$y_t = \text{argmin}_y \sum_{f \in F} \ell_{f,t}(x_t, y). \tag{6}$$

First, we claim that $(x_1, y_1), ..., (x_T, y_T)$ are "legal", in the sense that there is at least one potential target function g such that $g(x_1) = y_1, ..., g(x_T) = y_T$ for which there is a $h \in F$ with $d(g,h) \leq \mu(h)$. To see this, note that (4), (6), and (5) imply

$$\begin{aligned} \sum_{f \in F} \sum_{t=1}^{T} \ell_{f,t}(x_t, y_t) &= \sum_{t=1}^{T} \sum_{f \in F} \ell_{f,t}(x_t, y_t) \\ &< \left(\sum_{f \in F} (1 + \mu(f)) \right) - \phi(F, \mu) \qquad (7) \\ &\leq \left(\sum_{f \in F} (1 + \mu(f)) \right). \end{aligned}$$

Thus, there is a $h \in F$ such that $\sum_{t=1}^{T} \ell_{h,t}(x_t, y_t) \leq \mu(h)$. So if g is defined by $g(x_1) = y_1, ..., g(x_T) = y_T$ and $g(x) = h(x)$ for $x \notin \{x_1, ..., x_T\}$, g satisfies the requirements of a target function.

Suppose A outputs h. Loosely speaking, any $f \in F$ that is too far from h had better be eliminated as a possible candidate by

$$(x_1, y_1), ..., (x_T, y_T),$$

since otherwise an adversary could choose f to be close to the target. Specifically, for any $f \in F$ such that $d(h,f) > \mu(h) + \mu(f)$, it must be the case that $\sum_{t=1}^{T} \ell_{f,t}(x_t, y_t) > \mu(f)$; if

$$\sum_{t=1}^{T} \ell_{f,t}(x_t, y_t) \leq \mu(f)$$

then the definition of $\ell_{f,t}$ implies that

$$\sum_{t=1}^{T} \ell(f(x_t), y_t) \leq \mu(f),$$

and an adversary could modify f to get a target function g with distance at most $\mu(f)$ from f. Since $\mu(h) + \mu(f) < d(f,h) \leq d(f,g) + d(g,h) \leq \mu(f) + d(g,h)$, we get $d(g,h) > \mu(h)$ contradicting the fact that A outputs h. Thus

$$\begin{aligned}
\sum_{f \in F} \sum_{t=1}^{T} \ell_{f,t}(x_t, y_t) &\geq \sum_{f \in F : d(f,h) > \mu(f) + \mu(h)} (1 + \mu(f)) \\
&\geq \left(\sum_{f \in F} (1 + \mu(f)) \right) - \sum_{f \in F : d(f,h) \leq \mu(f) + \mu(h)} (1 + \mu(f)) \\
&\geq \left(\sum_{f \in F} (1 + \mu(f)) \right) - \phi(F, \mu),
\end{aligned}$$

contradicting (7) and completing the proof. □

Now we're ready for our theorem about $B_{F,\eta}$.

THEOREM 6. *Choose X, a set F of functions from X to Y, and an integer $\eta \geq 1$. Then*

$$Q(B_{F,\eta}, F, \eta) \leq 2\mathrm{opt}(F,\eta) \ln(|F|/(1+\deg(F,\eta))) + 2(\eta+1)(1+\deg(F,\eta)).$$

Proof: Consider a run of algorithm $B_{F,\eta}$ in which it asks queries $x_1, ..., x_T$, which are answered with $y_1, ..., y_T$. For each t, let

$$F_t = \left\{ f_{X - \{x_1, ..., x_{t-1}\}} : f \in F, \sum_{s<t} \ell(f(x_s), y_s) \leq \mu(f) \right\}.$$

We divide our analysis of $B_{F,\eta}$ into two stages. Let

$$S = \max \left\{ t : \sum_{f \in F_t} (1 + \mu_t(f)) \geq 2(\eta+1)(1 + \deg(F,\eta)) \right\}.$$

Note that for all $t \leq S$, $\sum_{f \in F_t} (1 + \mu_t(f)) \geq 2(\eta+1)(1 + \deg(F,\eta))$ as both $\mu_t(f)$ and $|F_t|$ decrease with t.

Let $t \leq S$. Since F_{t+1} is a set consisting of restrictions of elements of F_t, each subset of functions in F_t that is mapped by the restriction to the same function in F_{t+1} contributes only one value of $\mu_{t+1}(f)$ (where

$\mu_{t+1}(f) = \mu_t(f) - \ell'_{f,t}(x_t, y_t))$ to the sum $\sum_{f \in F_{t+1}} (1 + \mu_{t+1}(f))$. This gives

$$\sum_{f \in F_{t+1}} (1 + \mu_{t+1}(f)) \leq \left(\sum_{f \in F_t} (1 + \mu_t(f)) \right) - \sum_{f \in F_t} \ell'_{f,t}(x_t, y_t).$$

Lemma 5 implies that the choice of x_t chosen by $B_{F,\eta}$ will then give

$$\begin{aligned} &\sum_{f \in F_{t+1}} (1 + \mu_{t+1}(f)) \\ &\leq \left(\sum_{f \in F_t} (1 + \mu_t(f)) \right) \qquad (8) \\ &\qquad - \frac{1}{\mathrm{opt}(F_t, \mu_t)} \left(\left(\sum_{f \in F_t} (1 + \mu_t(f)) \right) - \phi(F_t, \mu_t) \right) \qquad (9) \end{aligned}$$

We will now prove that

$$\phi(F_t, \mu_t) \leq (\eta + 1)(1 + \deg(F, \eta)). \qquad (10)$$

For each $f \in F_t$, let $f^E \in F$ be obtained by extending f to X so as to minimize $\sum_{s<t} \ell(f^E(x_s), y_s)$. Recall that

$$\phi(F_t, \mu_t) = \max_{h \in F_t} \sum_{f \in F_t : d(f,h) \leq \mu_t(f) + \mu_t(h)} (1 + \mu_t(f)).$$

Choose $h_* \in F_t$ achieving this maximum. We have

$$\begin{aligned} (\eta + 1)(1 + \deg(F, \eta)) &= (\eta + 1) \max_{h \in F} |\{f \in F : d(f, h) \leq 2\eta\}| \\ &\geq (\eta + 1) |\{f \in F : d(f, h_*^E) \leq 2\eta\}| \\ &= \sum_{f \in F : d(f, h_*^E) \leq 2\eta} (\eta + 1) \\ &\geq \sum_{f \in F_t : d(f^E, h_*^E) \leq 2\eta} (\eta + 1). \qquad (11) \end{aligned}$$

For any $f, h \in F_t$,

$$\begin{aligned} d(f^E, g^E) &= \sum_{x \in X} \ell(f^E(x), h^E(x)) \\ &\leq d(f, h) + \sum_{s<t} \ell(f^E(x_s), h^E(x_s)) \\ &\leq d(f, h) + \sum_{s<t} \ell(f^E(x_s), y_s) + \ell(h^E(x_s), y_s) \\ &\leq d(f, h) + 2\eta - (\mu_t(f) + \mu_t(h)) \end{aligned}$$

since, by definition, for all $f \in F_t$, $\sum_{s<t} \ell(f^E(x_s), y_s) \leq \eta - \mu_t(f)$. The last inequality follows from the fact that $\mu_t(f) \leq \mu(f)$ for all t and F_t is a subset of F that is restricted to $X - \{x_1, \ldots, x_{t-1}\}$.

Thus (11) implies

$$\begin{aligned}(\eta+1)(1+\deg(F,\eta)) &\geq \sum_{f\in F_t : d(f,h_*) \leq \mu_t(f)+\mu_t(h_*)} (\eta+1) \\ &\geq \sum_{f\in F_t : d(f,h_*) \leq \mu_t(f)+\mu_t(h_*)} (1+\mu_t(f)),\end{aligned}$$

proving (10).

Putting (10) together with (9) and using $(\eta+1)(1+\deg(F,\eta)) \leq \sum_{f\in F_t}(1+\mu_t(f))/2$ from the definition of S, we have

$$\begin{aligned}\sum_{f\in F_{t+1}} (1+\mu_{t+1}(f)) &\leq \left(\sum_{f\in F_t}(1+\mu_t(f))\right) - \frac{1}{2\mathrm{opt}(F_t,\mu_t)} \sum_{f\in F_t}(1+\mu_t(f)) \\ &= \left(1 - \frac{1}{2\mathrm{opt}(F_t,\mu_t)}\right) \sum_{f\in F_t}(1+\mu_t(f)) \\ &\leq \left(1 - \frac{1}{2\mathrm{opt}(F,\mu)}\right) \sum_{f\in F_t}(1+\mu_t(f)). \qquad (12)\end{aligned}$$

The second inequality holds because $\mathrm{opt}(F_t,\mu_t) \leq \mathrm{opt}(F,\mu)$, which can be seen as follows. Choose an optimal algorithm $A_{\mathrm{opt},F,\mu}$ for learning F with quota function μ. We will construct an algorithm A' for learning F_t with μ_t using $A_{\mathrm{opt},F,\mu}$ and $\{(x_1,y_1),\ldots,(x_{t-1},y_{t-1})\}$. By definition, $\mathrm{opt}(F_t,\mu_t)$ is no more than the worst case number of queries required by A'. We now describe A' and show that it gives a correct solution using at most $\mathrm{opt}(F,\mu)$ queries, implying that $\mathrm{opt}(F_t,\mu_t) \leq \mathrm{opt}(F,\mu)$.

Algorithm A' simulates $A_{\mathrm{opt},F,\mu}$, answering some of $A_{\mathrm{opt},F,\mu}$'s queries, and passing others on to its oracle. If $A_{\mathrm{opt},F,\mu}$ asks x_s for $s < t$, then A' answers y_s. Any other of $A_{\mathrm{opt},F,\mu}$'s queries are asked by A', which then passes the answers back to $A_{\mathrm{opt},F,\mu}$. When $A_{\mathrm{opt},F,\mu}$ halts and returns its target, A' returns its restriction to $X - \{x_1,...,x_{t-1}\}$. Consider the run of A' in which the target function is $g' : X - \{x_1,...,x_{t-1}\} \to Y$. Define $g : X \to Y$ by setting $g(x_i) = y_i$ for each of $x_1,...,x_{t-1}$, and $g(x) = g'(x)$ for all other $x \in X$. Since g' is a target function, there must be a function $f' \in F_t$ such that $d(g',f') \leq \mu_t(f')$; choose such an f'. The definition of μ_t implies that there is an $f \in F$ such that

$$\begin{aligned}&\mu_t(f') = \mu(f) - \left(\sum_{s=1}^{t-1} \ell(f(x_s), y_s)\right) \\ &\forall x \in X - \{x_1,...,x_{t-1}\}, f(x) = f'(x);\end{aligned} \qquad (13)$$

choose such an f. We have

$$\begin{aligned} d(g,f) &= \sum_{x\in X} \ell(g(x),f(x)) \\ &= \left(\sum_{x\in X-\{x_1,\ldots,x_{t-1}\}} \ell(g(x),f(x))\right) + \sum_{s=1}^{t-1} \ell(g(x_s),f(x_s)) \\ &= d(g',f') + \sum_{s=1}^{t-1} \ell(g(x_s),f(x_s)) \\ &\leq \mu_t(f') + \sum_{s=1}^{t-1} \ell(g(x_s),f(x_s)) \\ &= \mu(f), \end{aligned}$$

by (13). Since there is an $f \in F$ such that $d(g,f) \leq \mu(f)$, and A' simulates answers to queries consistent with g, algorithm $A_{\mathrm{opt},F,\mu}$ returns a function h satisfying $d(g,h) \leq \mu(h)$ after asking at most $\mathrm{opt}(F,\mu)$ queries. Let h' be the function returned by A'; i.e., h' is the restriction of h to $X - \{x_1,...,x_{t-1}\}$. Then the following hold, where the last step is due to the definition of μ_t:

$$\begin{aligned} d(g',h') &= d(g,h) - \sum_{s=1}^{t-1} \ell(h(x_s),g(x_s)) \\ &\leq \mu(h) - \sum_{s=1}^{t-1} \ell(h(x_s),g(x_s)) \\ &\leq \mu_t(h'). \end{aligned}$$

Thus A' outputs a function h' with $d(g',h') \leq \mu_t(h')$ after asking at most $\mathrm{opt}(F,\mu)$ queries. Since g' was an arbitrarily chosen target for learning F_t with quota function μ_t, we have $\mathrm{opt}(F_t,\mu_t) \leq \mathrm{opt}(F,\mu)$, which implies (12).

Inequality (12) in turn implies

$$\sum_{f\in F_S} (1+\mu_S(f)) \leq \left(1 - \frac{1}{2\mathrm{opt}(F,\eta)}\right)^S |F|(\eta+1).$$

But, by definition, $\sum_{f\in F_S}(1+\mu_S(f)) \geq 2(\eta+1)(1+\deg(F,\eta))$, and so

$$\begin{aligned} 2(\eta+1)(1+\deg(F,\eta)) &\leq \left(1 - \frac{1}{2\mathrm{opt}(F,\eta)}\right)^S |F|(\eta+1) \\ 2(\eta+1)(1+\deg(F,\eta)) &\leq \exp\left(-\frac{S}{2\mathrm{opt}(F,\eta)}\right)|F|(\eta+1) \\ S &\leq 2\mathrm{opt}(F,\eta)\ln(|F|/(1+\deg(F,\eta))). \end{aligned}$$

For all $t \leq T$, since $|F_t| > 1$, and $\ell(u,v) \geq 1$ for $u \neq v$,

$$\sum_{f \in F_{t+1}} (1 + \mu_{t+1}(f)) \leq \left(\sum_{f \in F_t} (1 + \mu_t(f)) \right) - 1. \tag{14}$$

Since $\sum_{f \in F_{S+1}} (1 + \mu_{S+1}(f)) < 2(\eta+1)(1 + \deg(F, \eta))$ and $\sum_{f \in F_T} (1 + \mu_T(f)) \geq 1$, (14) implies that $T - S \leq 2(\eta+1)(1 + \deg(F, \eta))$. This completes the proof. □

4. Conclusion

We have described a theoretical treatment of the problem of how to choose questions to ask users in a collaborative filtering system. The shortcomings of our initial effort present a number of questions for future research.

Our analysis referred to the "η-degree" of a collection of experts. Is there a polynomial-time algorithm with a similar bound, depending logarithmically on $|F|$, for classes F of unbounded η-degree? What if the requirement that the output of the algorithm be at a distance η from the target is relaxed to allow $O(\eta)$?[1]

Can an algorithms that handle missing values be analyzed similarly? One step in this direction would involve relaxing the requirement that ℓ is a metric.

Do any ideas from the design of the algorithms of this paper have practical utility?

Can any of the ideas of this paper be profitably applied to the related problem of decision tree induction?

5. Acknowledgements

Wee Sun Lee and Phil Long gratefully acknowledge the support of National University of Singapore Academic Research Fund grant R252–000–070–107. We would like to thank the anonymous referees for pointing out several errors in earlier versions of the paper.

References

Angluin, D.: 1988, 'Queries and Concept Learning'. *Machine Learning* **2**, 319–342.

[1] Avrim Blum posed this question.

Angluin, D.: 2001, 'Queries revisited'. *Proceedings of the Twelfth International Conference on Algorithmic Learning Theory* pp. 12–31.

Angluin, D., M. Krikis, R. H. Sloan, and G. Turán: 1997, 'Malicious omissions and errors in answers to membership queries'. *Machine Learning* **28**, 211–255.

Arkin, E. M., H. Meijer, J. S. B. Mitchell, D. Rappaport, and S. Skiena: 1998, 'Decision trees for geometric models'. *International Journal of Computational Geometry and Applications* **8**(3), 343–364.

Arya, S., D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu: 1994, 'An optimal algorithm for approximate nearest neighbor searching'. *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms* pp. 573–582.

Breese, J., D. Heckerman, and C. Kadie: 1998, 'Empirical analysis of predictive algorithms for collaborative filtering'. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* pp. 43–52.

Bshouty, N. H., R. Cleve, R. Gavalda, S. Kannan, and C. Tamon: 1996, 'Oracles and Queries That Are Sufficient for Exact Learning'. *Journal of Computer and System Sciences* **52**(3).

Feige, U.: 1998, 'A Threshold of $\ln n$ for Approximating Set Cover'. *Journal of the ACM* **45**(4), 634–652.

Gortler, S. and R. Servedio: 2001, 'Quantum versus Classical Learnability'. In: *Sixteenth Conference on Computational Complexity (CCC).* pp. 138–148.

Hegedus, T.: 1995, 'Generalized teaching dimensions and the query complexity of learning'. In: *Proceedings of the 1995 Conference on Computational Learning Theory.* pp. 108–117.

Hellerstein, L., V. Raghavan, K. Pillaipakkamnatt, and D. Wilkins: 1996, 'How many queries are needed to learn?'. *Journal of the ACM* **43**(5), 840–862.

Hyafil, L. and R. L. Rivest: 1976, 'Constructing Optimal Binary Decision Trees is NP-Complete'. *Information Processing Letters* **5**(1), 15–17.

Indyk, P. and R. Motwani: 1998, 'Approximate nearest neighbors: Towards removing the curse of dimensionality'. *Proceedings of the 30th ACM Symposium on the Theory of Computing* pp. 604–613.

Johnson, D. S.: 1974, 'Approximation algorithms for combinatorial problems'. *Journal of Computer and System Sciences* **9**, 256–278.

Kushilevitz, E., R. Ostrovsky, and Y. Rabani: 1998, 'Efficient search for approximate nearest neighbor in high dimensional spaces'. *Proceedings of the 30th ACM Symposium on the Theory of Computing* pp. 614–623.

Nakamura, A. and N. Abe: 1998, 'Collaborative Filtering using Weighted Majority Prediction Algorithms'. In: *Proceedings of the Fifteenth International Conference on Machine Learning.*

Resnick, P., N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl: 1994, 'Grouplens: An open architecture for collaborative filtering of netnews'. In: *Proceedings of the ACM 1994 Conference on Computer Supported Cooperative Work.*

Resnick, P. and H. R. Varian: 1997, 'Recommender systems'. *Communications of the ACM* **40**, 56–58.

Seung, H. S., M. Opper, and H. Sompolinsky: 1992, 'Query by committee'. *Proceedings of the 1992 Workshop on Computational Learning Theory* pp. 287–294.