Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

ELSEVIER

Information Sciences 151 (2003) 171–200

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

Guided inference of nested monotone Boolean functions

Vetle I. Torvik [a,*], Evangelos Triantaphyllou [b]

[a] *Department of Psychiatry, University of Illinois at Chicago, MC 912, 1601 W Taylor Street, Chicago, IL 60612, USA*

[b] *Department of Industrial and Manufacturing Systems Engineering, Louisiana State University, 3128 CEBA Building, Baton Rouge, LA 70803-6409, USA*

Received 10 July 2002; received in revised form 3 December 2002; accepted 1 January 2003

Abstract

This paper addresses the problem of minimizing the average query complexity of inferring a pair of nested monotone Boolean functions defined on $\{0,1\}^n$ using a pair of oracles. Here, nested refers to the case when one of the functions is always greater than or equal to the other function. It is shown that the nested case is equivalent to inferring the single function case defined on $\{0,1\}^{n+1}$ when access to the two oracles is unrestricted. Two common examples of restricted oracles, namely sequential oracles and a single three-valued oracle, are also analyzed. The most efficient known approach to minimizing the average query complexity in inferring a single monotone Boolean function is based on a query selection criterion. It is shown that the selection criterion approach is easily modified for use with restricted oracles. Several real world examples illustrate the necessity and sufficiency of the nested monotone Boolean function model. Extensive computational results indicate that the nestedness assumption reduces the average query complexity by a few percent. This is a dramatic improvement considering the fact that this complexity is exponential in n.

Keywords: Nested monotone Boolean functions; Membership queries; Guided inference; Partially ordered sets (posets); Query selection criteria; Average query complexity

* Corresponding author.
E-mail addresses: vtorvik@uic.edu (V.I. Torvik), trianta@lsu.edu (E. Triantaphyllou).
URLs: http://arrowsmith2.psych.uic.edu/torvik, http://www.imse.lsu.edu/vangelis.

0020-0255/03/$ - see front matter
doi:10.1016/S0020-0255(03)00062-8

1. Introduction

Unlocking the mystery of natural phenomena is arguably a universal objective in scientific research. The rules governing a phenomenon can most often be learned by observing it under a sufficiently high number of conditions that are sufficiently high in resolution. This paper presents an active (or guided) approach to this knowledge discovery task. The underlying motivation for this approach is that being able to observe the phenomenon under specifically selected conditions may increase the accuracy and completeness of the learned knowledge at a faster rate than a passive observer who may not receive the pieces relevant to the puzzle soon enough. This active learning environment can be thought of as learning by successively submitting queries to an oracle which responds with a Boolean value (phenomenon is present or absent). In practice, the oracle may take the shape of a human expert, or it may be the outcome of performing tasks such as running experiments or searching large databases.

This paper focuses on applications where a given set of predictor variables have monotonically non-decreasing effects on the phenomenon under study. For example, a college applicant's grade point average is likely to have a positive effect on being accepted into a particular college when all other factors are constant. As another example, suppose a personal computer tends to crash when it runs a particular web browser and word processor simultaneously. Then, it will probably crash if it also runs its CD player.

This paper further focuses on situations where the monotone functions and their common n variables are Boolean (i.e., take on two values, say 0 and 1). This does not necessarily limit the application domain as Kovalerchuk et al. [12] establishes that any function can be represented by a sequence of monotonically non-increasing and non-decreasing Boolean functions. Recent literature contains a plethora of fields where monotone Boolean functions appropriately capture the phenomenon under study. Such diverse fields include, but are not limited to, social workers' decisions, lecturer evaluation and employee selection [1], chemical carcinogenicity, tax auditing and real estate valuation [3], breast cancer diagnosis and engineering reliability [13], signal processing [17], rheumatology [2], social sciences [9], finance [15], and record linkage in databases [10].

In applications, the monotonicity property is usually easy to identify due to its intuitive nature. This is perhaps its most important feature when human interaction is involved, since people tend to make very good use of knowledge they can interpret, understand, and validate. Monotonicity is a property that not only is sufficient and necessary in many applications but can also make the knowledge discovery process easier and more efficient. Breast cancer, college acceptance policies, and record linkage in databases are just a few applications that we use to demonstrate these points.

A single monotone Boolean function does not capture the idea of a value intermediate to 0 and 1, while a pair of nested monotone Boolean functions does so. Here the functions f_1 and f_2 are referred to as nested when the relationship $f_1 \geqslant f_2$ (or $f_1 \leqslant f_2$) holds. For example, in classification problems some instances might belong to a class with a high probability (i.e., where $f_2 = 1$ and $f_1 = 1$), and some might belong to the other class with a high probability (i.e., where $f_2 = 0$ and $f_1 = 0$). Other instances might not be classifiable with a satisfactorily high probability. A pair of nested functions allows for an intermediate classification (i.e., where $f_2 = 0$ and $f_1 = 1$) to be incorporated. This fact makes the monotone Boolean function model more powerful. The applications of breast cancer diagnosis, record linkage for databases and college applicant evaluations, described in Sections 3.2, 3.4 and 3.6, respectively, illustrate that more complex models are not necessary and simpler models are not sufficient.

In practice, a great deal of effort is put into the process of inferring functions underlying certain phenomena. Software is tested to establish its reliability, clinical trials are performed to establish the effect of drugs on diseases, etc. This inference process generally involves gathering and analyzing data. Gathering the data often involves some sort of labor that far outweighs the computations used to analyze the data in terms of cost. Therefore, the main objective in this paper is to minimize the labor associated with gathering the data (inquiries to oracles), as long as it is computationally feasible.

Monotone Boolean functions lay the ground for a simple question-asking strategy where it may be easy to pinpoint questions whose answers make incomplete knowledge more general or stochastic knowledge more accurate. More specifically, monotone Boolean functions are reconstructed by successive and systematic function evaluations (*membership queries* submitted to an oracle). The *oracle* can be thought of as an entity that knows the underlying monotone Boolean function and provides a Boolean value in response to each query. Initially, the entire set of 2^n Boolean vectors, denoted by $\{0,1\}^n$, is considered unclassified. A vector is then selected from the set of unclassified vectors and is submitted to the oracle as a membership query. After the vector's function value is provided by the oracle, the set of unclassified vectors is reduced. These queries are then repeated until all of the vectors are classified. Due to the underlying monotonicity and nestedness properties, a learning algorithm that actively chooses its observations (i.e., is guided) may significantly increase the learning rate, as a passive learner might not receive the relevant pieces of information fast enough. Therefore, it is highly desirable not only to be able to pose questions (or queries), but to pose "smart" questions.

The main problem addressed in this paper is how to identify these "smart" questions using query selection criteria in order to completely and efficiently infer a pair of functions underlying some phenomenon by only knowing that

they belong to the class of nested monotone Boolean functions defined on at most n variables. It is assumed that the learning algorithm has access to a pair of oracles that know the underlying functions, and each oracle provides a 0 or 1 function value in response to each query. The problem when the function to be inferred is just a single monotone Boolean function has been studied extensively in the literature (e.g., [4,6,7,16,18,22]). A comparative study of existing algorithms for and a new query selection criterion approach to this problem can be found in [20]. An extension to a stochastic oracle, in particular when the oracle may misclassify each query, has been addressed in [21]. A unified and more detailed version of these papers can be found in [19].

This paper differs from existing work by analyzing nested monotone Boolean functions and the corresponding ways oracles may be queried in practice, and has four major points. First, it shows how a selection criterion approach to minimizing the average query complexity is extended to three different inference scenarios ((1) sequentially from two oracles, (2) simultaneously from a single oracle, and (3) simultaneously from two oracles) pertaining to a pair of nested monotone Boolean functions. Second, it demonstrates how the nested monotone Boolean function model often is sufficient (i.e., a more complex model is not necessary) and necessary (i.e., simpler models are not sufficient) for a wide variety of real world applications. Third, it quantifies the reduction in average query complexity due to the nestedness assumption. Fourth, it compares the query complexities for the three different inference scenarios.

This paper is organized as follows. Section 2 provides some background information on monotone Boolean functions, and briefly describes the related inference problems. Section 3 presents the details of the inference problems, and illustrates the solution methodology on real world examples. The concluding remarks given in Section 5 are based on the computational results presented in Section 4.

2. Background information

The purpose of this section is to provide some background information on monotone Boolean functions and the related inference problems. Section 2.1 addresses monotone Boolean functions and the extension to a pair of nested functions. Section 2.2 defines the inference problems and their respective objectives.

2.1. Some properties of monotone Boolean functions

Let $\{0,1\}^n$ denote the set of Boolean vectors defined on n Boolean variables. A deterministic Boolean function defined on $\{0,1\}^n$ is simply a mapping to

$\{0,1\}$. A vector v in $\{0,1\}^n$ is said to *precede* another vector w, denoted by $v \preceq w$, if and only if (iff) $v_i \leqslant w_i$ for $i = 1, 2, \ldots, n$. Here v_i (and w_i) denotes the ith element of vector v (and w, respectively). When a vector v precedes another vector w and the two vectors are distinct (i.e., $v \neq w$,), then v is said to *strictly precede* w, denoted by $v \prec w$. A monotone Boolean function f is called *non-decreasing* if $f(v) \leqslant f(w) \forall (v,w) : v \preceq w$, and *non-increasing* if $f(v) \geqslant f(w) \forall (v,w) : v \preceq w$. The set of non-decreasing functions defined on $\{0,1\}^n$ is denoted by M_n. This paper focuses on non-decreasing functions, which are referred to as *monotone*, as analogous results hold for non-increasing functions.

A vector v^* is called an *upper zero* of a function f if $f(v) > f(v^*) \forall v : v^* \prec v$. Similarly, a vector v^* is called a *lower unit* of a function f if $f(v) < f(v^*) \forall v : v \prec v^*$. Lower units and upper zeros are also referred to as *border vectors*. For any monotone Boolean function f, the set of lower units LU(f), and the set of upper zeros, UZ(f) are unique and either one of these two sets uniquely identifies f.

A pair of monotone Boolean functions f_1 and f_2 are called *nested* when the following relationship holds: $f_1(v) \geqslant f_2(v)$ (or $f_1(v) \leqslant f_2(v)$), $\forall v \in \{0,1\}^n$. The case when $f_1 \geqslant f_2$ is addressed in this paper as analogous results hold for the case when $f_1 \leqslant f_2$. In other words, if $f_2(v)$ is equal to 1, then $f_1(v)$ must also be equal 1, and if $f_1(v)$ is equal to 0, then $f_2(v)$ must also be equal 0, for any vector v. The latter definition gives meaning to the word nested, while the more succinct expression $f_1 \geqslant f_2$ will be used throughout this paper.

The number of pairs of nested monotone Boolean functions defined on $\{0,1\}^n$ is simply $\Psi(n+1)$, where $\Psi(n)$ denotes the number of monotone Boolean functions defined on $\{0,1\}^n$. This fact can be observed by constructing the partially ordered set (or poset for short) connecting two posets $P_1 = (\{0,1\}^n, \preceq)$ and $P_2 = (\{0,1\}^n, \preceq)$ associated with functions f_1 and f_2 respectively, by adding the edges corresponding to the precedence relations $f_1(v) \geqslant f_2(v)$, $\forall v \in \{0,1\}^n$. Posets can be viewed as directed graphs where each vertex corresponds to a vector and a directed edge from vertex v to vertex w, represents the precedence relation $v \preceq w$. The operation of constructing the new poset can be viewed as *poset multiplication*, denoted by $\otimes$, where the domain $\{0,1\}^n$ makes up a poset and the functions make up another poset $\{0,1\}$. Multiplying $\{0,1\}^n$ by $\{0,1\}$ creates $\{0,1\}^{n+1}$, as the example for $n = 2$ in Fig. 1 shows.

Poset multiplication can be generalized to any set of vectors V and any relation between the functions $f_1, f_2, \ldots, f_m$. As an example consider the set of vectors $V = \{1,2,3\}$, and three functions that satisfy the following relationships $f_1 \leqslant f_2 \leqslant f_3$. Here, the set of vectors makes up a chain poset of length 3, and the functions make up a chain poset of length 3. The new poset created by multiplying the two posets is a 3×3 matrix poset. This paper focuses on the case when $V = \{0,1\}^n$ and $F = \{(0,0),(0,1),(1,1)\}$, while the methodology is applicable to any sets $V \subset \mathbb{R}^n$ and $F \subset \mathbb{R}^r$ as long as they are both finite.

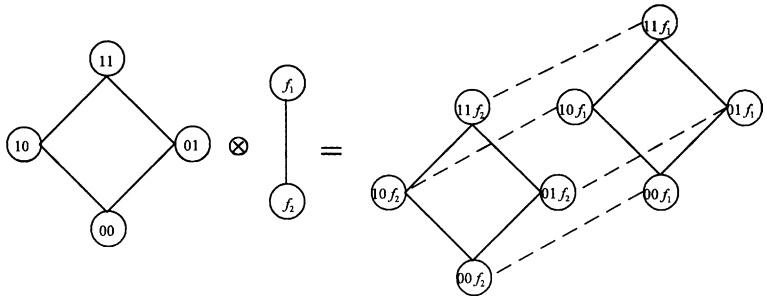

Fig. 1. Illustration of poset multiplication.

2.2. Guided inference

Monotone Boolean functions lay the ground for a simple question-asking strategy, which forms the basis of this paper. More specifically, the problem of uncovering monotone Boolean functions by successive and systematic *function evaluations* (*membership queries* submitted to an oracle) is addressed. This paper deals with the problem of inferring a pair of nested functions with access to two oracles (one for each function). The manner in which the functions' values are obtained (i.e., restrictions on the access to oracles) depends on the application. This problem is further broken down into Problems #1, #2 and #3 (described in Sections 3.1, 3.3 and 3.5, respectively) by the manner in which the oracles are accessed. For each of these three problems, real world applications are described in Sections 3.2, 3.4 and 3.6, respectively. The wide variety of applications demonstrates the versatility of the monotonicity and nestedness assumptions and illustrates the three major types of formulations they can take on.

Problem #1 deals with applications where the costs associated with queries to one of the oracles far outweighs the costs associated with the queries to the other oracle, and situations where one has sequential access to the oracles. In such applications, it is beneficial to reconstruct the function associated with the least expensive oracle first, after which the reconstruction of the other function begins. In other words, two monotone Boolean functions are to be sequentially inferred from two oracles. Problem #2 deals with applications where a single oracle knows both functions' values. That is, the oracle answers each query with a pair of function values $(f_1, f_2) = (0, 0)$, (1,0), or (1,1). In other words, two monotone Boolean functions are to be simultaneously inferred from a single three-valued oracle. For this problem the two nested monotone Boolean functions are viewed as a single monotone function f taking on the three values 0, 1 and 2, corresponding to $(f_1, f_2) = (0, 0)$, (1,0) and (1,1), respectively. Notice that (f_1, f_2) cannot take on the values (0,1) due to the nestedness

constraint $f_1 \geqslant f_2$. The single three-valued function is used to emphasize that the Boolean function values arrive in pairs, for each vector, from a single oracle. Problem #3 deals with applications where the costs associated with access to the two oracles are the same, and situations that allow for switching between the two oracles intermittently. At any inference step of such applications, it does not matter which oracle is queried. In other words, two monotone Boolean functions are to be simultaneously inferred with unrestricted access to two oracles.

The inference process consists of the following steps. Initially, the entire set of 2^n Boolean vectors $\{0,1\}^n$, is considered to be unclassified for both functions f_1 and f_2. That is, the values of underlying monotone Boolean functions f_1 and f_2 are all unknown and may be 0 or 1. A vector v is then selected from the set of vectors U_i that are unclassified by function f_i and is submitted to oracle i as a membership query (either $i = 1$ or 2). After the vector's function value $f_i(v)$ is provided by the oracle, the set of unclassified vectors U_i is reduced according to the following monotonicity constraints: $f_i(w) = 0$, $\forall w \in U_i$: $w \preceq v$, when $f_i(v) = 0$, or the following monotonicity constraints: $f_i(w) = 1$, $\forall w \in U_i$: $v \preceq w$, when $f_i(v) = 1$, for $i = 1$ or 2. Similarly, the other set of unclassified vectors (i.e., U_1 if $i = 2$, and U_2 if $i = 1$) may be reduced according to the following nestedness constraints: $f_1(w) = 1$, $\forall w \in U_1$: $v \preceq w$, when $f_2(v) = 1$, or the following nestedness constraints: $f_2(w) = 0$, $\forall w \in U_2$: $w \preceq v$, when $f_1(v) = 0$. Vectors are then repeatedly selected from the unclassified set until all of the vectors are classified by both functions (i.e., $U_1 = U_2 = \{\}$).

Given the classification of any vector in $\{0,1\}^n$, some of the other vectors may be concurrently classified if the underlying functions are assumed to be monotone. Therefore, only a subset of the 2^n vectors may need to be evaluated in order to completely reconstruct the underlying functions. A key problem is then to select "promising" vectors so as to reduce the total number of queries (or *query complexity*). In practice, the queries are often associated with some sort of effort, such as consulting with experts, performing experiments or running simulations. In these situations, the cost of the queries far exceeds the computational cost. This paper is, therefore, focused on minimizing the query complexity as long as it is computationally feasible.

Kovalerchuk et al. [13] considered the problem of inferring a pair of nested monotone Boolean functions. Their algorithm is an extension of Hansel's algorithm [7] for a single monotone Boolean function. It exhibited a promising efficiency in their cancer diagnosis application. However, their performance analysis is far from conclusive as a single application represents a single pair of nested monotone Boolean functions.

For the single monotone Boolean function case, Torvik and Triantaphyllou [20] suggested minimizing the average number of queries over the entire class of monotone Boolean functions defined on $\{0,1\}^n$. This objective can be expressed mathematically as follows:

$$Q(n) = \min_{A} \frac{\sum_{f \in M_n} \varphi(A, f)}{\psi(n)},$$

where $\varphi(A, f)$ denotes the number of queries performed by algorithm A in reconstructing the monotone Boolean function f. The analogous objective for nested monotone Boolean functions is used in this paper. The minimum average number of queries for Problem # k (for $k = 1$, 2, and 3) can be expressed mathematically as follows:

$$Q_k(n) = \min_{A_k} \frac{\sum_{f_1, f_2 \in M_n : f_2 \leqslant f_1} \varphi(A_k, f_1, f_2)}{\psi(n+1)},$$

where $\varphi(A_k, f_1, f_2)$ denotes the number of queries performed by algorithm A_k, in reconstructing the pair of nested monotone Boolean functions f_1 and f_2. Here A_1, A_2 and A_3 denote algorithms designed for Problems #1, #2 and #3, respectively. The details of these three problems and the respective inference algorithms are provided in Section 3.

Since the three problems differ in the way the oracles are queried, it should be clarified that a query unit pertains to the membership value from one of the two functions f_1 and f_2. This definition is intuitive for Problems #1 and #3, where two oracles are accessed individually. For Problem #2, however, the membership values are provided in pairs from a single three-valued oracle. To make the definition of $Q_2(n)$ comparable to $Q_1(n)$ and $Q_3(n)$, each access to the three-valued oracle in Problem #2 will be counted as two queries.

The minimum average number of queries for the unrestricted problem $Q_3(n)$ is equal to that of the single function case in one dimension higher $Q(n+1)$. To see this connection consider a pair of nested monotone Boolean functions f_1 and f_2 defined on $\{0,1\}^n$. The query domain for the nested case can be viewed as the product: $\{0,1\}^n \times \{f_1, f_2\}$. Each of the vertices in the resulting poset $(\{0,1\}^{n+1}, \preceq)$, may take on function values of 0 or 1, where the monotonicity property is preserved. In other words, a pair of nested monotone Boolean functions defined on $\{0,1\}^n$, are equivalent to a single monotone Boolean function defined on $\{0,1\}^{n+1}$.

As an example, consider inferring a pair of nested monotone Boolean functions defined on $\{0,1\}^2$. Fig. 1 shows the query domain in the form of the poset $(\{0,1\}^3, \preceq)$, where the vertices are labeled (vf_i), for $i = 1$ and 2. Each vertex in this poset is in the form of a query, where a sample query now could be $(01f_1)$. If the answer to that query is 0, then the vertices strictly preceding $(01f_1)$ (i.e., $(00f_1)$,$(01f_2)$, and $(00f_2)$) in the new poset are also assigned the value 0. This leaves the vertices $(11f_1)$, $(10f_1)$, $(11f_2)$, $(10f_2)$ as unclassified.

As a side note, consider a set of functions making up a poset of the form $(\{0,1\}^m, \preceq)$ where each function is defined on the set of vectors of the form $(\{0,1\}^n, \preceq)$. The query domain can then be viewed as a poset of the form

$(\{0,1\}^n, \preceq) \otimes (\{0,1\}^m, \preceq) = (\{0,1\}^{n+m}, \preceq)$. For example, inferring four functions from four unrestricted oracles, defined on $\{0,1\}^5$, satisfying the relations $f_1 \leqslant f_2, f_1 \leqslant f_3, f_2 \leqslant f_4$ and $f_3 \leqslant f_4$, is equivalent to inferring a single monotone Boolean function defined on defined on $\{0,1\}^{2+5} = \{0,1\}^7$.

3. The three key inference problems and some applications

This section describes the three key inference problems in detail. The goal of the three problems is the same, namely to minimize the average number of queries used to infer a pair of nested monotone Boolean functions defined on the poset $\{0,1\}^n$. However, the manner in which the oracles that provide the function values are accessed differs between the three problems.

In [20] we showed that the selection criterion $\min |K_1(v) - K_0(v)|$ minimized the average query complexity $Q(n)$ for a single monotone Boolean function when $n \leqslant 4$, and was close to optimal when $n \geqslant 5$. Here, $K_z(v)$ denotes the number of vertices that are concurrently classified when $f(v) = z$, for $z = 0$ and 1. This selection criterion will be used for the three nested problems with slight modifications.

The query domain for the nested case is made up of the set of vectors $\{0,1\}^n \times \{f_1, f_2\}$. For a vertex labeled (vf_i), let $K_z(v, f_i)$ be the number of vertices that are concurrently classified when the value of $f_i(v)$ is queried and the answer is $f_i(v) = z$, for $z = 0$ and 1. When the oracles are unrestricted (i.e., Problem #3), vertices are selected based on the criterion $\min |K_1(v, f_i) - K_0(v, f_i)|$. This criterion is equivalent to the criterion $\min |K_1(v) - K_0(v)|$ for the single function case. The only change is in the notation since the oracle that is to provide the answer has to be identified for Problem #3. This criterion is, therefore, referred to as $\min |K_1 - K_0|$. For sequential oracles (i.e., Problem #1), queries of the form $f_2(v)$ are infeasible until all of the queries of the form $f_1(v)$ are classified. In this case, the criterion used during the first phase is $\min |K_1(v, f_1) - K_0(v, f_1)|$, after which the criterion $\min |K_1(v, f_2) - K_0(v, f_2)|$ is used. For the three-valued oracle (i.e., Problem #2), the queries are of the form $(f_1(v), f_2(v))$ and are selected using the criterion $\min |K_{11}(v) - K_{00}(v)|$. Here, the value of the function $K_{zz}(v)$ equals the number of vertices concurrently classified when vertex v is queried and the result of the query is $f_1(v) = f_2(v) = z$, for $z = 0$ and 1. Once there are no pairs of vertices of the form $(f_1(v), f_2(v))$ left unclassified, the criterion $\min |K_1(v, f_i) - K_0(v, f_i)|$ is used for the remaining of the query selections.

3.1. Problem # 1—sequentially inferring nested functions

For this problem, the two functions are considered to be available via their two respective oracles where the inference situation dictates that, for example,

function f_1 should be completely reconstructed before the inference of function f_2 begins. In other words, the two functions are to be *sequentially inferred*. This approach may simply be the only feasible or reasonable one or it may be dictated by the cost of querying the oracle associated with f_2 far surpassing the cost of querying the other oracle.

3.2. An application of Problem #1 to breast cancer diagnosis

Breast cancer diagnosis will be used to illustrate the sequential inference problem. Fig. 2 shows the two step sequential process in which breast cancer is diagnosed. In the first step, an X-ray image (called a mammogram) of a tumor is studied by a radiologist who determines whether performing a biopsy is necessary. If the radiologist finds that a biopsy is not necessary, the diagnosis is concluded. Otherwise, a biopsy is performed (a surgeon removes small piece of the tumor) and a pathologist studies the removed tissue under a microscope to see whether it is cancerous. One can argue that the cost of the invasive biopsy procedure and analysis outweighs the cost of analyzing a mammogram and that patients (if not surgeons) are reluctant to undergo a biopsy procedure without a radiologist recommendation. Therefore, the radiologist can be viewed as the oracle governing function f_1 which should be completely restored first. The biopsy can be viewed as the oracle governing function f_2 whose restoration begins afterwards. The nature of the two functions f_1 and f_2 implies that they are nested (i.e., $f_1 \geqslant f_2$).

To illustrate the inference process using the criterion $\min |K_1 - K_0|$ for sequential inference, consider the pair of nested monotone Boolean functions inferred by interviewing a radiologist in [13]. The inferred function that describes their "biopsy subproblem" is defined as follows:

$$f_1(v) = v_1v_2 \vee v_3 \vee v_1v_4 \vee v_2v_4 \vee v_5,$$

where $f_1(v) = 1$ if a biopsy is recommended for a tumor with the features described by v. The inferred function that describes their "cancer subproblem" is defined as follows:

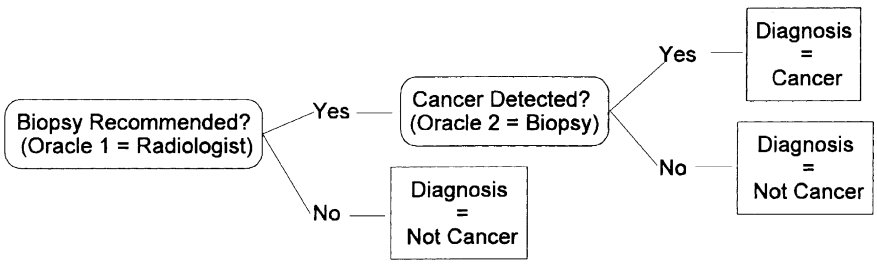

Fig. 2. Illustration of sequential oracles in breast cancer diagnosis.

$$f_2(v) = v_1v_2 \vee v_3 \vee v_1v_5 \vee v_2v_5 \vee v_4v_5,$$

where $f_2(v) = 1$ if a tumor with the features described by v is highly suspicious for malignancy and 0 otherwise. Here v_i describes the ith diagnostic feature which is 1 if it is "pro-cancer" and 0 if it is "contra-cancer". The five individual features are defined as follows:

$v_1 = 1$ if *amount and volume of calcifications* is "pro-cancer", 0 otherwise,
$v_2 = 1$ if *shape and density of calcifications* is "pro-cancer", 0 otherwise,
$v_3 = 1$ if *ductal orientation* is "pro-cancer", 0 otherwise,
$v_4 = 1$ if *comparison with previous exam* is "pro-cancer", 0 otherwise and
$v_5 = 1$ if *associated findings* is "pro-cancer", 0 otherwise.

It should be noted that v_1 and v_2 are also monotone Boolean functions that have to be inferred prior to the inference of f. The details of that decomposition are left out for the purpose of simplifying this illustration. The interested reader is referred to [13,14] for more details. It should also be noted that the two functions were both inferred from querying the radiologist. For the purpose of illustrating the sequential oracles, it is assumed here that the values of function f_2 are available via the biopsy procedure. It is also assumed that a set of mammograms and their features is available for selection, and this set covers all vectors $\{0,1\}^5$.

Fig. 3 shows the sequence of queries and their answers for the breast cancer diagnosis application, when the queries are selected based on the criterion $\min |K_1 - K_0|$. Below each query, the remaining unclassified vectors are shown in the form of a poset with the selected vector circled. Initially, all of the vectors $\{0,1\}^5 \otimes \{f_2, f_1\} = \{0,1\}^6$ are unclassified and several vectors on the middle layer possess the same minimum feasible value $|K_1 - K_0| = 0$. One of these vectors $(000111) = (00011f_1)$ is arbitrarily selected for the first query. The radiologist is asked whether a biopsy is recommended for a tumor with "pro-cancer" features in *comparison with previous examination* ($v_4 = 1$) and in *associated findings* ($v_5 = 1$). After the radiologist answers "yes" (i.e., $f_1(00011) = 1$), the set of unclassified vectors is reduced and then forms the poset shown below the second query. In this poset, the minimum $|K_1 - K_0|$ value of 1, belongs to the vector $(111000) = (11100f_2)$ which is an infeasible query at this point. However, there are several vectors with the minimum feasible $|K_1 - K_0|$ value of 2. The vector $(011001) = (01100f_1)$ is one of them. The radiologist is asked whether a biopsy is recommended for a tumor with "pro-cancer" features in *shape and density of calcifications* ($v_2 = 1$) and in *ductal orientation* ($v_3 = 1$). After the radiologist answers "yes" (i.e., $f_1(01100) = 1$), the set of unclassified vectors is further reduced and forms the poset shown below the third query. This process continues and after the 10th query, the biopsy recommendation function f_1 is completely restored (i.e., all of the vectors of the form $(v_1v_2v_3v_4v_51)$ are classified).

At this point the inference of the cancer function f_2 from the biopsy oracle begins. The feasible vectors are now of the form $(v_1v_2v_3v_4v_50)$, and make up the

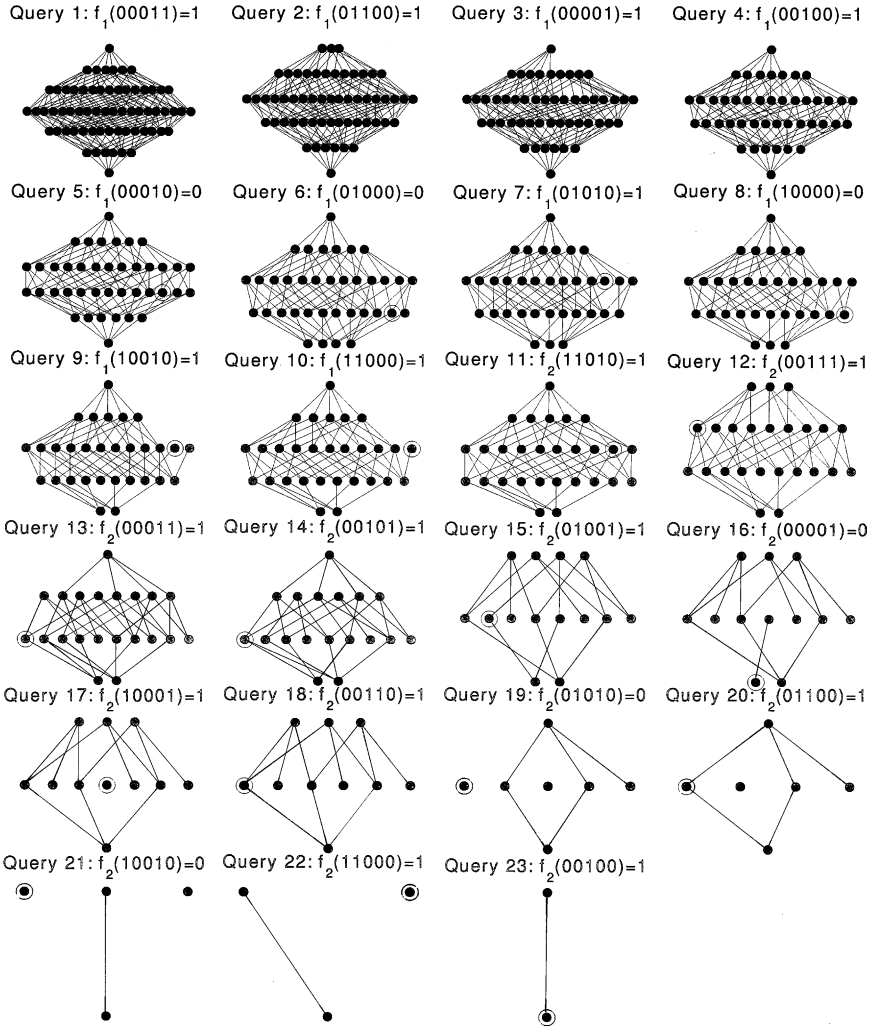

Fig. 3. Sequential inference of the two nested breast cancer diagnosis functions using the selection criterion.

poset shown below the 11th query. The minimum $|K_1 - K_0|$ value of 0 belongs to vector $(110100) = (11010f_2)$. A biopsy is performed on a tumor with "pro-cancer" features in *amount and volume of calcifications* ($v_1 = 1$), in *shape and density of calcifications* ($v_2 = 1$), and in *comparison with previous examination* ($v_4 = 1$). After the cancer is detected (i.e., $f_2(11010) = 1$), the set of unclassified vectors is further reduced and forms the poset shown below the 12th query.

The querying continues for a total of 23 queries; the first 10 for the radiologist and last 13 in the form of biopsies.

3.3. Problem #2—inferring nested functions from a single three-valued oracle

For this problem the two nested monotone Boolean functions are viewed as a single function f taking on the three values 0, 1 and 2, corresponding to $(f_1, f_2) = (0,0)$, (1,0) and (1,1), respectively. Notice that (f_1, f_2) cannot take on the values (0,1) due to the nestedness constraint $f_1 \geqslant f_2$. The single three-valued function is used to emphasize that the Boolean function values arrive in pairs, for each vector, from a single oracle.

This problem is similar to that of inferring a single monotone Boolean function in that a single oracle provides the answers. In contrast, some vectors may need to be evaluated by two oracles for Problems #1 and #3. In Problem #2, vectors are not considered just classified or unclassified, but also partially classified. Some time into the inference process unclassified vectors may take on the values 0, 1 and 2 as function values while partially classified vectors may take on the values 0 and 1, or the values 1 and 2.

3.4. An application of Problem #2 to record linkage in databases

The problem of merging a pair of databases with n common fields will be used to illustrate a three-valued oracle. Fig. 4 shows a database administrator as an oracle who is asked whether a pair of records should be merged. The goal in the record linkage problem is to find which records in Database A matches the records in Database B so that the newly created database does not have replications of the same records nor is missing any of the records from either of the two old databases. The interested reader is referred to [5,10,23] for further details on the record linkage problem.

Suppose the two databases have n common fields. Let variable v_i be defined as 1 if a pair of records from the two databases have the same value on the ith field, and 0 otherwise, for $i = 1, 2, \ldots, n$. Consider a Database A defined with the following fields: *first name, last name, state of residence, email address, age, gender, salary*, and a Database B defined on the following fields: *first name initial, middle name initial, last name, address, zip code, phone number,*

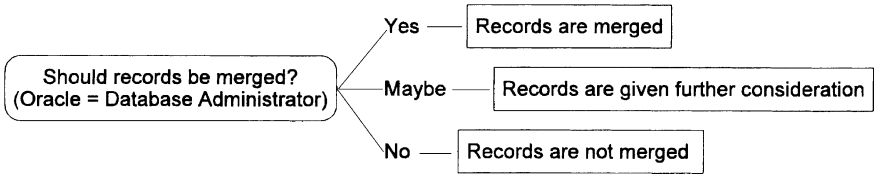

Fig. 4. Illustration of a single three-valued oracle in record linkage.

occupation, age, gender. For a pair of records r_A and r_B from Databases A and B, respectively, five variables can be defined by the five fields common to Databases A and B as follows:

$v_1 = 1$, if *first name initial*(r_A) = *first name initial*(r_B), 0 otherwise,
$v_2 = 1$, if *last name*(r_A) = *last name*(r_B), 0 otherwise,
$v_3 = 1$, if *state of residence*(r_A) = *state of residence*(r_B), 0 otherwise,
$v_4 = 1$, if *age*(r_A) = *age*(r_B), 0 otherwise, and
$v_5 = 1$, if *gender*(r_A) = *gender*(r_B), 0 otherwise.

Please notice that the state of *zip code*(r_B) can be converted into *state of residence*(r_B), and *first name initial*(r_A) can be extracted from *first name*(r_A).

Suppose that a pair of records should be merged if they have the same values on at least:

(*first name initial*, *last name*, *state of residence* and *age*), or
(*first name initial*, *last name*, *state of residence* and *gender*), or
(*first name initial*, *last name*, *age* and *gender*), or
(*last name*, *state of residence*, *age* and *gender*).

Further, suppose that a pair of records should be given further consideration if they do not satisfy the conditions for merging and have same values on at least:

(*first name initial*, *last name* and *state of residence*), or
(*first name initial*, *last name* and *age*), or
(*last name*, *age* and *gender*).

That is, the underlying functions to be inferred are given as follows:

$$f_1(v) = v_1v_2v_3 \vee v_1v_2v_4 \vee v_2v_4v_5, \quad \text{and}$$
$$f_2(v) = v_1v_2v_3v_4 \vee v_1v_2v_3v_5 \vee v_1v_2v_4v_5 \vee v_2v_3v_4v_5,$$

where the common values of a pair of records are described by the vector $v = (v_1v_2v_3v_4v_5)$. When $f_1(v) = f_2(v) = 0$, the records should not be merged. When $f_1(v) = 1$ and $f_2(v) = 0$, the records should be given further consideration. When $f_1(v) = f_2(v) = 1$, the records should be merged.

The inference process can be viewed as a dialogue between the database administrator and the person posing the queries. Fig. 5 shows the sequence of queries and their answers for the record linkage application, when the queries are selected based on the criterion $\min|K_{11} - K_{00}|$. Below each query, the remaining unclassified vectors are shown in the form of a poset with the selected vector(s) circled. Initially, all of the vectors $\{0,1\}^5 \otimes \{f_2, f_1\} = \{0,1\}^6$ are unclassified. A query of the form $f(v_1v_2v_3v_4v_5)$ corresponds to the pair of vectors $(v_1v_2v_3v_4v_50)$ and $(v_1v_2v_3v_4v_51)$. When there are no unclassified pairs left, the queries are given in the form $f_1(v_1v_2v_3v_4v_5)$ or $f_2(v_1v_2v_3v_4v_5)$ to show what information was gained by the query $f(v_1v_2v_3v_4v_5)$.

Initially, several pairs of vectors possess the same minimum feasible value $|K_{11} - K_{00}| = 8$. One of these pairs $\{(000110), (000111)\}$ is arbitrarily selected

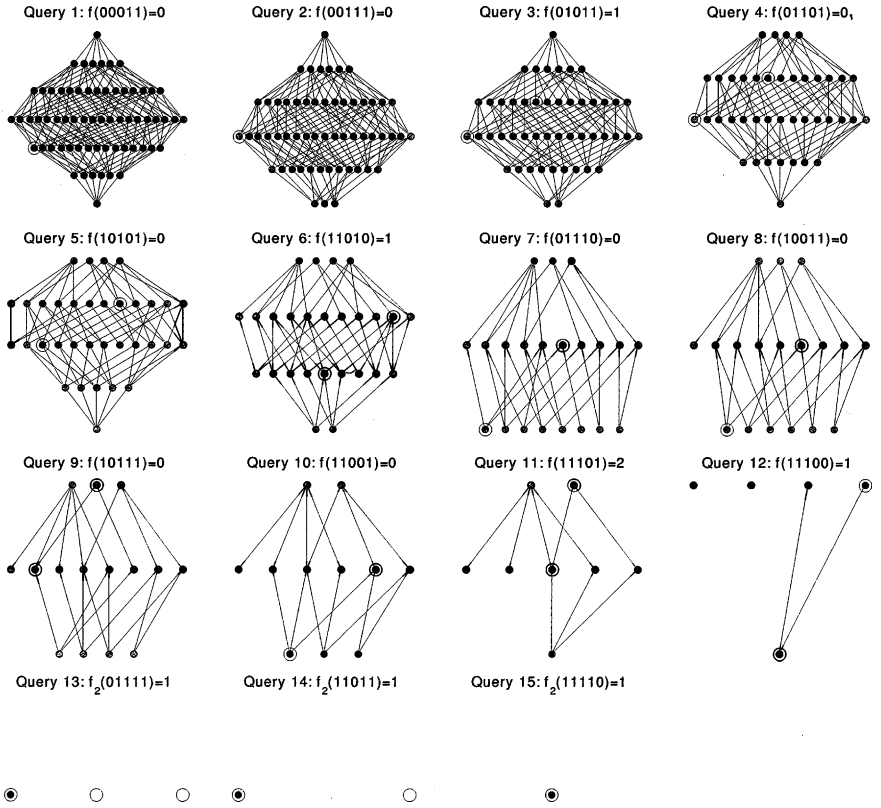

Fig. 5. Inferring the two nested record linkage functions from a three-valued oracle using the selection criterion.

for the first query. The database administrator is asked whether two records with the same *age* ($v_4 = 1$) and *gender* ($v_5 = 1$) should be merged. After the database administrator answers "no" (i.e., $f(00011) = 0$), the set of unclassified vectors is reduced and then forms the poset shown below the second query. In this poset, the minimum $|K_{11} - K_{00}|$ value of 0, belongs to the pair of vectors $\{(001110), (001111)\}$. The database administrator is asked whether two records with the same *state of residence* ($v_3 = 1$), *age* ($v_4 = 1$) and *gender* ($v_5 = 1$) should be merged. After the database administrator answers "no" (i.e., $f(00111) = 0$), the set of unclassified vectors is reduced and then forms the poset shown below the third query. This process continues and after the 13th query, there are no pairs of unclassified vectors of the form $\{(v_1v_2v_3v_4v_50), (v_1v_2v_3v_4v_51)\}$.

From there on the selection criterion $\min |K_1 - K_0|$ is used. From Fig. 5, it is obvious that the remaining vectors (011110), (110110) and (111100) are

unrelated. As a result, all of them have to be queried and the query order does not matter. For the 13th query, the database administrator is asked whether two records with the same *last name* ($v_2 = 1$), *state of residence* ($v_3 = 1$), *age* ($v_4 = 1$) and *gender* ($v_5 = 1$) should be merged. After the database administrator answers "yes" (i.e., $f(01111) = 2$), the set of unclassified vectors is reduced and then forms the poset shown below the 14th query. After 15 queries to the database administrator, the record linkage function is completely restored. Please note that it is known from the first 12 queries that $f_1(01111) = f_1(11011) = f_1(11110) = 1$. In other words, the information gained by the 13th, 14th and 15th queries are that $f_2(01111) = f_2(11011) = f_2(11110) = 1$, respectively, which are shown over the respective posets in Fig. 5.

3.5. Problem #3—inferring nested functions from two unrestricted oracles

This problem is similar to Problem #1, in that two oracles are queried separately. Unlike Problem #1, no restrictions are put on the manner in which the two oracles are queried. At each inference step, a vector can be submitted to any of the two oracles. In this sense, this is the least restrictive of the three problems, and it is therefore expected that this approach will be the more efficient.

3.6. An application of Problem #3 to college applicant evaluations

Consider the evaluation process used for accepting students into a particular college illustrated in Fig. 6. When a selection committee evaluates applications they often place them into three ordered categories based on variables such grade point average (GPA), standardized test scores, quality of an essay, recommendations, etc. The bottom category consists of students who are not to be accepted. The top category consists of students who are to be accepted, while an intermediate category consists of applicants who need to be considered more carefully.

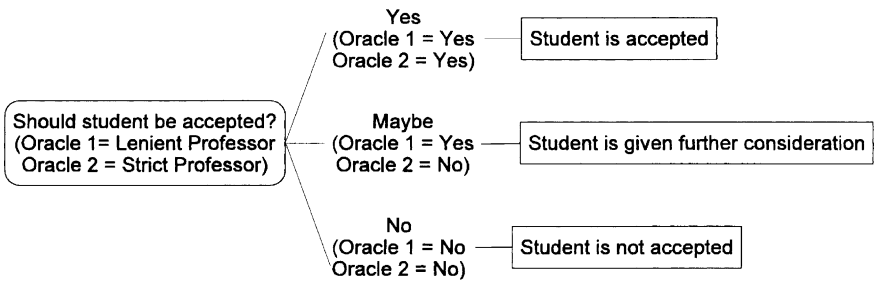

Fig. 6. Illustration of two oracles for the student acceptance policy.

Suppose a college dean wishes to establish guidelines that the selection committee is to follow when placing applicants into these three categories. To that end, the dean asks two professors from the college, one who is known to be lenient and another who is known to be strict, to evaluate applications. The lenient professor's acceptance function is given by f_1 while the strict professor's acceptance function is given by f_2. Here, $f_1(v) = 1$ if a student with criteria described by vector v is accepted by the lenient professor, and $f_1(v) = 0$ if it is not accepted by the lenient professor. Also, $f_2(v) = 1$ if a student with criteria described by vector v is accepted by the strict professor, and $f_2(v) = 0$ if it is not accepted by the strict professor.

It is assumed that a student with criteria v, who is accepted by the strict professor will also be accepted by the lenient professor (i.e., $f_2(v) = 1$ implies that $f_1(v) = 1$). It is also assumed that, if the student is not accepted by the lenient professor, the student will neither be accepted by the strict professor (i.e., $f_1(v) = 0$ implies that $f_2(v) = 0$). This describes the nestedness assumption made of the two professors acceptance functions: $f_1 \geqslant f_2$.

The college dean has two oracles (the two professors) that govern the two functions, where the order in which the oracles are queried is not restricted. That is, any application can be evaluated by either of the two professors at any given time during the inference process.

For the purpose of illustrating the inference process consider the following four simplified acceptance criteria for the college as follows:

$v_1 = 1$ for *a high GPA*, 0 otherwise,

$v_2 = 1$ for *a high GRE score*, 0 otherwise,

$v_3 = 1$ for *an essay of high quality*, 0 otherwise and

$v_4 = 1$ for *good recommendations*, 0 otherwise.

Suppose, the lenient professor will accept students with the following minimum requirements:

(*a high GPA*), or

(*a high GRE score* and *good recommendations*), or

(*an essay of high quality* and *good recommendations*),

while the strict professor will accept students with the following minimum requirements:

(*a high GPA*, *a high GRE score* and *good recommendations*), or

(*a high GPA*, *an essay of high quality* and *good recommendation*).

That is, the two functions to be inferred are:

$$f_1(v) = v_1 \vee v_2v_4 \vee v_3v_4, \quad \text{and} \quad f_2(v) = v_1v_2v_4 \vee v_1v_3v_4.$$

The students who satisfy the strict and the lenient professors' minimum requirements will be accepted. The students who satisfy the lenient professor's minimum requirements but not the strict professors are given further consideration. The students who do not satisfy either of the professors' minimum requirements are not accepted.

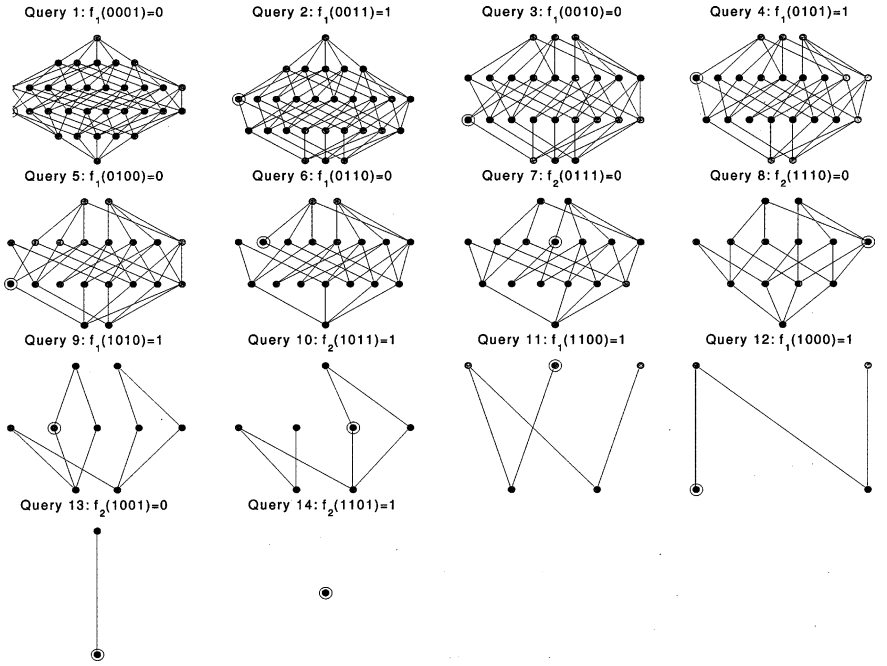

Fig. 7. Simultaneous inference of the two nested college acceptance policy functions using the selection criterion.

Fig. 7 shows the sequence of queries and their answers for the student evaluation application when the queries are selected based on the criterion $\min |K_1 - K_0|$. Below each query, the remaining unclassified vectors are shown in the form of a poset with the selected vector circled. Initially, all of the vectors $\{0,1\}^4 \otimes \{f_2, f_1\} = \{0,1\}^5$ are unclassified. A vector of the form $(v_1v_2v_3v_41)$ corresponds to a query of the form $f_1(v_1v_2v_3v_4)$, and a vector of the form $(v_1v_2v_3v_40)$ corresponds to a query of the form $f_2(v_1v_2v_3v_4)$.

Several vectors on the two middle layers possess the same minimum feasible value $|K_1 - K_0| = 4$. One of these vectors $(00011) = (0001f_1)$ is arbitrarily selected for the first query. The lenient professor evaluates whether a student with *good recommendations* ($v_4 = 1$) should be accepted. After the lenient professor answers "no" (i.e., $f_1(0001) = 0$), the set of unclassified vectors is reduced and then forms the poset shown below the second query. In this poset, the minimum $|K_1 - K_0|$ value of 0 belongs to the vectors (00111), (01011) and (10011). The vector $(00111) = (0011f_1)$ is selected. The lenient professor evaluates whether a student with *an essay of high quality* ($v_3 = 1$) and *good recommendations* ($v_4 = 1$) should be accepted. After the lenient professor answers "yes" (i.e., $f_1(0011) = 1$), the set of unclassified vectors is further reduced

and forms the poset shown below the third query. This process continues for an additional seven queries to the lenient professor and five queries to the strict professor (not in that order) as shown in Fig. 7. After the 14th query, both professors' acceptance functions, and hence the college acceptance policy, are completely restored.

3.7. Hierarchical decomposition of variables

The variables used for the different applications described in the previous sections are simplified in the illustrations. In some applications, the variables may be monotone Boolean functions defined on set of Boolean variables at a lower level. Kovalerchuk et al. [13] decomposed the five breast cancer diagnostic variables (from Section 3.2) in a hierarchical manner as follows. The first variable v_1 is defined as 1 if the *amount and volume of calcifications* is "pro-cancer", and 0 if it is "contra-cancer". In reality, this variable was inferred (through queries to the radiologist) as the following monotone Boolean function:

$$v_1(x_1, x_2, x_3) = x_2 \vee x_1 x_3.$$

Here, the extra variables are defined as follows:

$x_1 = 1$ if the *number of calcifications/cm*^{2} is "large", 0 if "small",
$x_2 = 1$ if the *volume of calcifications* (cm^3) is "small", 0 if "large" and
$x_3 = 1$ if the *total number of calcifications* is "large", 0 if "small".

The second variable v_2 is defined as 1 if the *shape and density of calcifications* is "pro-cancer", and 0 if it is "contra-cancer". In reality, this variable was inferred (through queries to the radiologist) as the following monotone Boolean function:

$$v_2(x_4, x_5, x_6, x_7, x_8) = x_4 \vee x_5 \vee x_6 x_7 x_8.$$

Here, the extra variables are defined as follows:

$x_4 = 1$ if the *irregularity in the shape of individual calcifications* is "marked", 0 if "mild",
$x_5 = 1$ if the *variation in the shape of calcifications* is "marked", 0 if "mild",
$x_6 = 1$ if the *variation in the size of calcifications* is "marked", 0 if "mild",
$x_7 = 1$ if the *variation in the density of calcifications* is "marked", 0 if "mild", and
$x_8 = 1$ if the *density of calcifications* is "marked", 0 if "mild".

In general, one can construct a hierarchy of the sets of variables, where each set of variables corresponds to an independent inference problem. Fig. 8 shows this hierarchy for the breast cancer diagnostic variables. The upper level consists of the set $\{v_1, v_2, v_3, v_4, v_5\}$ which is linked to the sets of variables $\{x_1, x_2, x_3\}$ and $\{x_4, x_5, x_6, x_7, x_8\}$ at the lower level. Here, the variables v_1 and v_2 have to be defined before the inference problem defined on the set variables

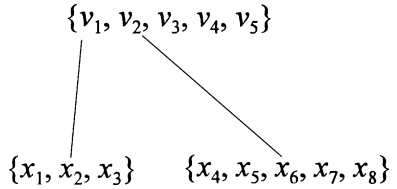

Fig. 8. Hierarchical decomposition of the variables for the breast cancer diagnosis application.

$\{v_1, v_2, v_3, v_4, v_5\}$ can begin. In general, the inference problems at the lower level have to be completed before the inference problems at the upper levels can begin.

The breast cancer inference problem is defined on the set of Boolean variables $\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, v_3, v_4, v_5, f_i\}$. This problem includes a total of $2^{12} = 4096$ vectors to choose from. However, it can be approached hierarchically, as three independent problems defined on the sets $\{x_1, x_2, x_3\}$, $\{x_4, x_5, x_6, x_7, x_8\}$, and $\{v_1, v_2, v_3, v_4, v_5, f_i\}$, respectively. These problems include a total of $2^3 + 2^5 + 2^6 = 104$ possible vectors to choose from. The hierarchical approach to this problem reduces the number of possible vectors to choose from by a factor of $4096/104 \approx 39.4$. Please notice that a single monotone Boolean function is to be inferred for each of the sets $\{x_1, x_2, x_3\}$, and $\{x_4, x_5, x_6, x_7, x_8\}$. This corresponds to the single function inference problem defined on the sets $\{0,1\}^3$ and $\{0,1\}^5$, respectively. In contrast, a pair of nested monotone Boolean functions defined on the set $\{v_1, v_2, v_3, v_4, v_5\}$ are to be sequentially inferred. This corresponds to Problem #1 and includes the query domain $\{0,1\}^6$.

Similar hierarchies can be constructed for the other applications described in this paper. As an example consider the college acceptance policy application described in Section 3.6. The variable corresponding to the GRE score may be defined in terms of the individual verbal, quantitative and analytical scores. That is, the variable v_1 may be defined as a monotone Boolean function defined on the following variables:

$x_1 = 1$ for *a high verbal GRE score*, 0 otherwise,

$x_2 = 1$ for *a high quantitative GRE score*, 0 otherwise and

$x_3 = 1$ for *a high analytical GRE score*, 0 otherwise.

For example, a quantitative score of 700 or higher (out of 800) may be considered as a high score, implying that the variable x_2 is equal to 1.

Similar decompositions can be created for the variables v_2, v_3 and v_4, as follows. The variable v_2 defined in terms of the recommendations may be a monotone Boolean function defined on the following variables:

$x_4 = 1$ if *highly motivated*, 0 otherwise,

$x_5 = 1$ if *creative*, 0 otherwise and

$x_6 = 1$ if *bright/smart*, 0 otherwise.

For example, if the people who wrote recommendations refer to the student as *highly motivated*, then the variable x_4 is equal to 1. The variable v_3 defined in terms of the essay may be a monotone Boolean function defined on the following variables:

$x_7 = 1$ for *good writing skills*, 0 otherwise,

$x_8 = 1$ if the *interests match the college program*, 0 otherwise and

$x_9 = 1$ for *a good vision*, 0 otherwise.

The variable v_4 defined in terms of the GPA may be a function defined on the following variables:

$x_{10} = 1$ for *a high science GPA*, 0 otherwise,

$x_{11} = 1$ for *a high arts GPA*, 0 otherwise and

$x_{12} = 1$ for *a high language GPA*, 0 otherwise.

Here, the variable x_{10} can be further decomposed into the following variables:

$y_1 = 1$ for *a high applied math GPA*, 0 otherwise,

$y_2 = 1$ for *a high theoretical math GPA*, 0 otherwise and

$y_3 = 1$ for *a high physics or chemistry GPA*, 0 otherwise.

Notice that the monotonicity assumption holds for each of the subsets of variables. For example, *a high applied math GPA* is more likely to result in *a high GPA*, than *a low applied math GPA*. Therefore, *a high applied math GPA* is more likely to result in acceptance than *a low math GPA*, since *a high GPA* is more likely to result in acceptance.

Fig. 9 shows the hierarchical decomposition of these variables. The inference problem defined on the set of Boolean variables $\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, y_1, y_2, y_3, x_{11}, x_{12}, f_i\}$ consists of $2^{15} = 32{,}768$ possible vectors to choose from. Fortunately, it can be approached hierarchically, as six independent problems. Five are in the form of the single function inference problem defined on the set $\{0,1\}^3$, and one is in the form of Problem #3 with the query domain $\{0,1\}^5$. Combined, these sets consist of $5 \times 2^3 + 2^5 = 72$ vectors to choose from. The hierarchical approach to this problem reduces the number of possible vectors to choose from by a factor of $32{,}768/72 \approx 455$.

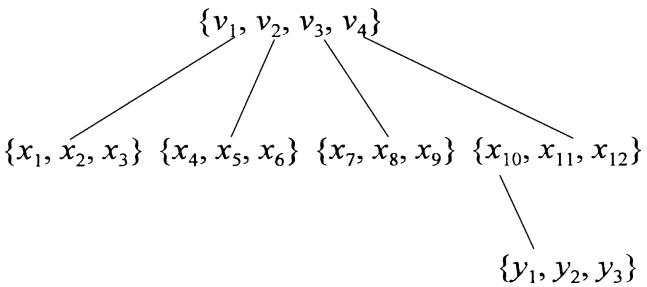

Fig. 9. Hierarchical decomposition of the variables for the college acceptance policy application.

As another illustrative example consider the record linkage application described in Section 3.4. Databases often contain errors. For example, the last name McGinniss may correspond to a person with the last name McGuinness. The variable v_2 is defined as 1 if a pair of last names are the same, and 0 otherwise. Hierarchical decomposition allows for similar last names to be matched, even when they are not an exact match. For example, the variable v_2 can be defined in terms of the following monotone Boolean function:

$$v_2(x_1, x_2) = x_1 x_2,$$

where the variables are given by:

$x_1 = 1$ if the *last names are unique*, 0 otherwise and

$x_2 = 1$ if *there is an agreement in a large number of the characters*, 0 otherwise.

For example, the last names McGinniss and McGuinness may result in $x_1 = 1$ and $x_2 = 1$, for which $v_3 = 1$. Notice here that the monotonicity assumption holds since the more unique (i.e., rarer) a last name is and the greater agreements in the number of characters, the more likely the last names are to be the same.

It should also be noted that for any of the inference problems considered in this paper, it is a strict requirement that function(s) at the uppermost level is (are) defined on a set of Boolean variables. Therefore, each set of variables at the lower level(s) has to correspond to a single Boolean function. Otherwise, one or more of the variables at higher levels will not be Boolean. As a consequence, a single two-valued oracle must be used for each of the inference problem(s) at the lower level(s) in Problems #1, #2 and #3.

In the breast cancer application it seems reasonable to use the radiologist as the oracle when inferring the functions at the lower level. In the college acceptance policy, it seems reasonable to consider the lenient and the strict professor together as a single oracle for the inference problems at the lower levels. That is, each query is answered with a Boolean value that is in agreement with both professors. For the record linkage application, the database administrator is the sole oracle who should provide Boolean answers in the inference problems at the lower level(s).

The reduction in the query domain is not the only benefit of the hierarchical decomposition of the inference problems. For example, a tumor with characteristics described by the 11 element vector $(x_1x_2x_3x_4x_5x_6x_7x_8v_3v_4v_5)$ may be hard for a radiologist to evaluate. In contrast, the queries for the three independent inference problems are of the much simpler forms $(x_1x_2x_3)$, $(x_4x_5x_6x_7x_8)$ and $(v_1v_2v_3v_4v_5)$. A college applicant given by the 14 element vector $(x_1x_2x_3x_4x_5x_6x_7x_8x_9y_1y_2y_3x_{11}x_{12})$ may be hard for the professors to evaluate. In contrast, the queries for the six independent inference problems are of the much simpler forms $(y_1y_2y_3)$, $(x_1x_2x_3)$, $(x_4x_5x_6)$, $(x_7x_8x_9)$, $(x_{10}x_{11}x_{12})$ and $(v_1v_2v_3v_4)$.

3.8. Randomly generating a pair of nested monotone Boolean functions

The previous sections showed the applicability of the nested monotone Boolean function inference and demonstrated how the various selection criteria can be used in practice. In this section, a framework for further analyzing the efficiency of this approach is developed. As discussed in [20], computing the exact average number of queries $Q(n)$ and identifying the corresponding optimal vertices for the single function problem become computationally burdensome for larger values of n. We therefore developed an algorithm for randomly generating monotone Boolean functions and estimating the average query complexity based on an unequal probability sampling framework. The same algorithm will be used here for generating nested monotone Boolean functions. The only difference is the manner in which the inclusion probabilities are computed. A brief overview of the algorithm is presented next, and the interested reader is referred to [20] for the details.

The algorithm creates the general Boolean function f by classifying $f(v)$ as 1 with probability $p(v)$, or 0 with probability $1-p(v)$ independently for all vectors in $\{0,1\}^n$. The first target monotone Boolean function f_1 is then defined by the lower units of f, while the second target monotone Boolean function f_2 is defined by the upper zeros of f. After the pair of nested functions have been generated, the inclusion probability has to be computed. To that end, define the two random monotone Boolean functions F_1 and F_2 by the output of a single execution of the algorithm. Then, the inclusion probability for the pair of functions (f_1, f_2) is simply the product of the individual vector assignment probabilities, excluding the vectors that lie in between the $\mathrm{LU}(f)$ and $\mathrm{UZ}(f)$, given by $S = \{0,1\}^n - \{w : w^1 \prec w \prec w^2, w^1 \in \mathrm{LU}(f), w^2 \in \mathrm{UZ}(f)\}$. The inclusion probability is given by the following equation:

$$\Pr\{F_1 = f_1 \vee F_2 = f_2\} = \prod_{v\in\{w:f(w)=1, w\in S\}} p(v) \prod_{v\in\{w:f(w)=0, w\in S\}} (1-p(v)).$$

To generate the functions close to uniformly, a random 0 or 1 vector assignment is performed according to the vector's respective fraction of monotone Boolean functions that classify it as 0 or 1. In [20] we developed the following definition:

$$p(v) = \frac{e^{\alpha(n,k)(k-n/2)}}{1+e^{\alpha(n,k)(k-n/2)}}, \quad v \in \{0,1\}^n \quad \text{and} \quad k = |v|,$$

where

$$\alpha(n,0) = \frac{\ln(\psi(n)-1)}{n/2}$$

and generalizing with the following approximation $\alpha(n,k) \approx \alpha(\max\{n-k,k\},0)$ for $k = 1,2,\ldots,n$.

It should be noted that it may be necessary to work with the inclusion probabilities on a logarithmic base as they get extremely small. For example, $\Psi(11) > 10^{144}$ using Korshunov's asymptotic [11]. This procedure does not result in complete uniformity, yet it is a step in this direction using independent assignments. For small values of n (less than 7 or 8), this approximation is not needed as the exact values are known.

4. Some results for inferring nested monotone Boolean functions

In [20] we developed the selection criterion $\min|K_1 - K_0|$ for the single monotone Boolean function inference problem that was optimal for the chain and the sawtooth posets of any size. This criterion was further observed to be optimal for $\{0,1\}^n$ when $n \leqslant 4$, and probably close to optimal for $\{0,1\}^n$ when $n > 4$. Since Problem #3 with n variables is equivalent to single monotone Boolean function inference problem with $n+1$ variables, the following relationship holds: $Q_3(n) = Q(n+1)$. As a result, the selection criterion $\min|K_1 - K_0|$ is optimal for Problem #3 with 3 or fewer variables and probably close to optimal for 4 or more variables.

The different selection criteria described in Section 3 do not specify which vector to select when there are ties. For the purpose of comparing the algorithms on the same ground and without introducing another aspect of randomness, ties were broken by selecting the first vector in the list of tied vectors. The results in Figs. 10–12 are based on an exhaustive analysis (i.e., all the monotone functions were generated) for n up to and including 4. For n greater than 4, random samples of functions were generated as described in Section 3.8. The number of pairs of nested monotone Boolean functions generated were 2000 for $n = 5$, 6, 7, and 200 for $n = 8$, 9, 10, and 100 for $n = 11$ and 12. This is the maximum number of pairs of functions used in the estimate, because the functions were generated with replacement. However, since the likelihood of generating the same functions more than once is small (especially for larger values of n) the effective sample size was generally close to these maxima.

Fig. 10 shows the average number of queries for the three problems when selection criteria are used. The Horvitz–Thompson [8] estimator was used to compute the averages for n greater than 4. The average number of queries is normalized by the maximum possible number of queries 2^{n+1} so that the magnitude of all the averages are not overshadowed by the large values obtained for n equal to 12. As a consequence, two algorithms that result in parallel curves in such a plot, have an exponential (in n) difference in the average number of queries. Also, the gap between the curves and the horizontal line $\textit{Average Number of Queries}/2^{n+1} = 1$ (not shown in the figure) can be thought of as the benefit of the monotone and nestedness assumptions to-

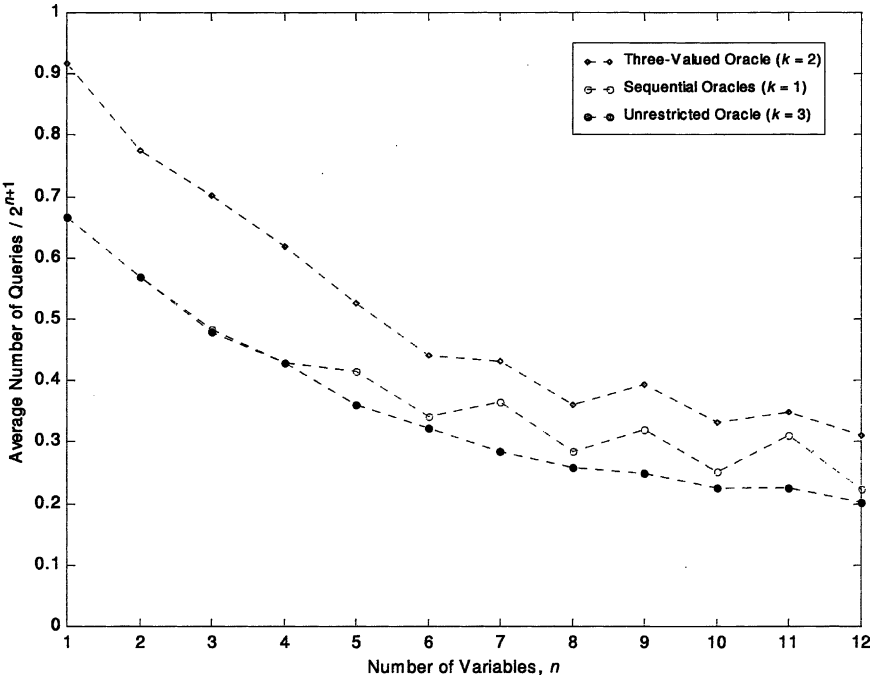

Fig. 10. Average query complexity of the selection criteria used for the poset $\{0,1\}^n$.

gether. This is due to the fact that 2^{n+1} is the number of required queries when the underlying pair of functions are not necessarily nested nor monotone.

The lower curve corresponds to the unrestricted case, which performed the fewest number of queries on the average. The selection criterion $\min |K_1 - K_0|$ used for the unrestricted problem achieves the minimum average number of queries for n up to 3. Its curve can therefore be thought of as a lower bound on the average number of queries for n up to 3. This curve seems to approach about 20% for $n \geqslant 12$, that is about 1600 queries out the maximum of $2^{13} = 8192$ are performed for $n = 12$. The middle curve corresponds to the sequential case. The sequential queries are not as efficient as the unrestricted queries, though they are very close for $n = 1$, 2, 3 and 4. The upper curve corresponds to the three-valued case, which is the least efficient of the three types of oracles.

Fig. 11 further quantifies the increase in the average number of queries due to the two restrictions on the oracles. As mentioned earlier, the sequential oracles are not very restrictive for $n = 1$, 2, 3 and 4, with the greatest increase in average number of queries at about 1% occurring for $n = 3$, and 0 for $n = 1$, 2 and 4. For $n \geqslant 5$, the increase in query complexity due to sequential oracles oscillates between two curves that increase with the number of variables. The

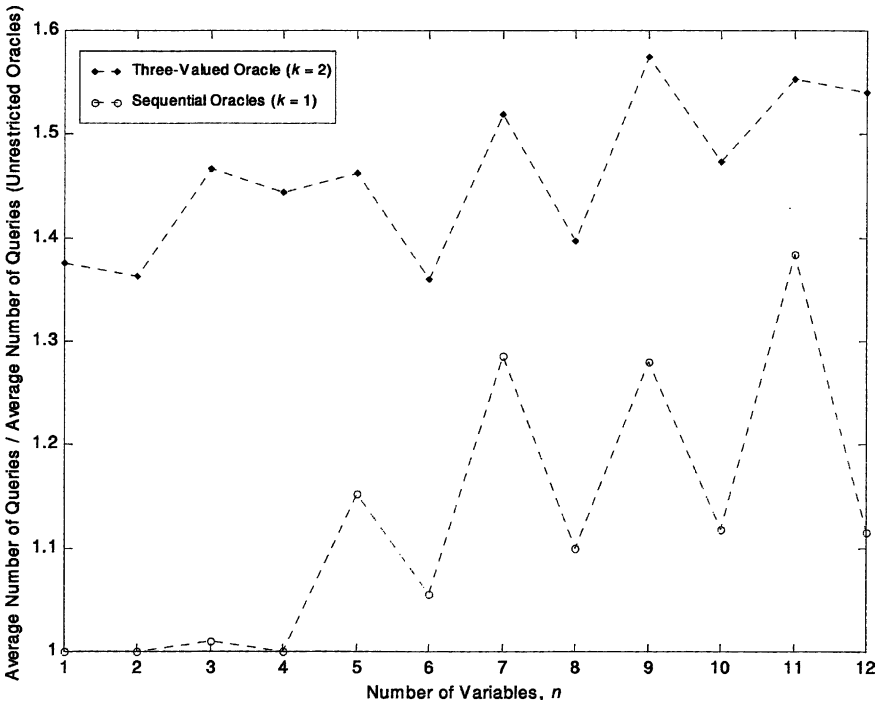

Fig. 11. Increase in average query complexities of the selection criteria on the poset $\{0,1\}^n$ due to restricted access to the oracles.

lower curve corresponds to the even number of variables and it levels off at about 12% for $n \geqslant 10$. The upper curve correspond to the odd number of variables and it levels off at about 33% for $n \geqslant 7$. That is, the increase in the average number of queries due to the sequential restriction is between 12% and 33% for $n \geqslant 7$.

In contrast, the three-valued oracle is much more restrictive. The increase in average number of queries due to the three-valued oracle, oscillates between 38% and 58%. The increases in the average number of queries for the sequential and three-valued cases are dramatic. This is due to the fact that the average number of queries increases exponentially with the number of variables.

Fig. 12 shows the reduction in the average number of queries due to the nestedness assumption. If the nestedness property of the two functions defined on $\{0,1\}^n$ is ignored, the minimum total number of queries is on the average $2Q(n)$. The benefit from the nestedness assumption for Problem #2 is quantified by the ratio of $Q_3(n)/2Q(n) = Q_3(n)/2Q_3(n-1)$. The values of $Q_3(n)$ were computed and shown for $n = 1, 2, \ldots, 12$ in Fig. 10. Here, $Q_3(0) = 5/3$ is used to compute the ratio $Q_3(1)/2Q_3(0)$. This reduction increases with the number

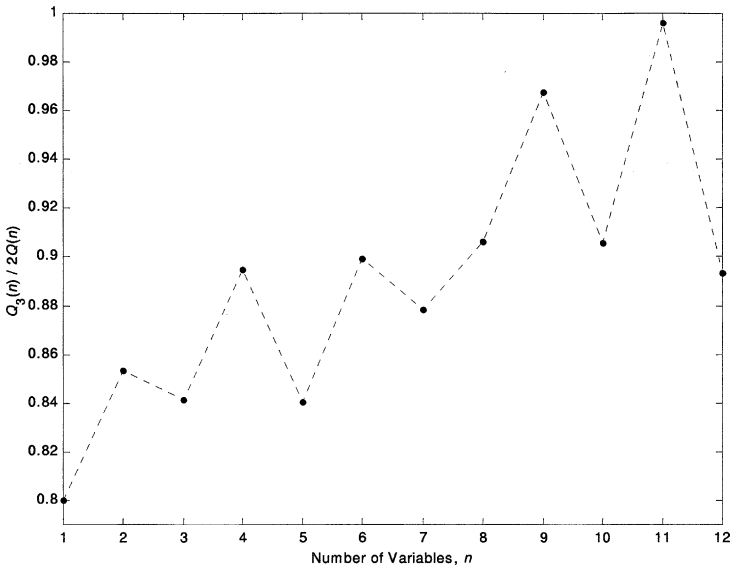

Fig. 12. Improvement in the average query complexity on the poset $\{0,1\}^n$due to the nestedness assumption.

of variables. It starts out at 20% for $n = 1$, and lies between 2% and 11% for $n > 7$.

5. Concluding remarks

This study is focused on the extension of the single monotone Boolean function inference problem to the inference of a pair of nested monotone Boolean functions. The benefits of this research are many-fold.

First, it shows how the optimal and selection criterion approach to minimizing the average query complexity is extended to three different inference scenarios pertaining to a pair of nested monotone Boolean functions. The selection criteria seem to be good choices for the nested inference problem. They are optimal for $n = 1$, 2, 3 and are probably very close to optimal for n greater than 3.

Second, it demonstrates how the nested monotone Boolean function model often is sufficient (i.e., a more complex model is not necessary) and necessary (i.e., simpler models are not sufficient) for a wide variety of real world applications. A simpler model, such as a single monotone Boolean function, will at best provide a poor approximation of the phenomenon under study. At worst, it will be unable to model the phenomenon. A more complex model, such as two independent monotone Boolean functions, will at the very least, result in

an increased query complexity. In addition, the inferred functions may lead to conflicting knowledge and are more likely to contain errors.

Third, it quantifies the reduction in average query complexity due to the nestedness assumption. The benefits of guided inference of a single monotone Boolean function using the selection criterion have been well documented in the past. The additional improvement due to the nestedness assumption is a few percent. However, the average query complexity on the poset $\{0,1\}^n$ is exponential in n, which makes a few percent reduction a dramatic improvement.

Fourth, it compares the efficiency of the three major types of oracles. The three-valued oracle provides the most significant restriction on the oracles. It causes up to 58% increase in the average number of queries. It is interesting to observe that the sequential oracles are just as efficient as the unrestricted oracles when the number of variables is less than five. This implies that the pair of nested monotone Boolean functions defined on these posets can be inferred sequentially without losing optimality for small problems.

The objective of minimizing the average query complexity based on the monotonicity and the nestedness assumptions is appropriate in a wide variety of real world applications. In some applications, additional properties may be known about the underlying function. In this case another objective may be more appropriate. For example, it may be that the application limits the number of lower units, shifting the focus of the optimal vertices from the vertical center to the vertical edge of the poset. In other cases, the cost of querying the first oracle may be less than, yet of similar magnitude as, the cost of querying the second oracle. Then, the first few queries should be directed at the first oracle. After a few queries it may be cost beneficial to also query the second oracle. It would be interesting to see how dialogue with the oracles change as the assumptions used in this paper are modified.

In this paper we focused on the query domain $\{0,1\}^n$. However, the query selection criterion approach to learning monotone Boolean functions is applicable in the much more general monotone setting: $V \to F$, where the sets $V \subset \mathbb{R}^n$ and $F \subset \mathbb{R}^r$ are both finite. The monotone mapping $V \to F$, where the set $V \subset \mathbb{R}^n$ is infinite and the set $F \subset \mathbb{R}^r$ is finite, forms another intriguing problem. It is well known that binary search is optimal when the query domain V is a bounded subset of the real line and $F = \{0,1\}$. However, when the set V is multidimensional and infinite (e.g., $V = [a,b]^2$), pinpointing the optimal queries is a much more complex problem. The selection criterion $\min |K_1 - K_0|$ can be modified to accommodate this case also. Let U denote the unclassified set (i.e., $\subset V$) and let the parameters $K_0(v)$ and $K_1(v)$ now denote the size of the subsets $\{w \in U : w \prec v\}$ and $\{w \in U : v \prec w\}$, respectively. For example, $K_z(v)$ is measured in terms of length, area, volume, etc. when $n = 1$, 2, 3, etc., respectively. The selection criterion $\min |K_1 - K_0|$ is then optimal for $n = 1$. How well this criterion performs when $n > 1$, is an open question.

Acknowledgement

The authors gratefully acknowledge the support from the Office of Naval Research (ONR) Grant N00014-97-1-0632.

References

[1] A. Ben-David, Automatic generation of symbolic multiattribute ordinal knowledge-based DSSs: methodology and applications, Decision Sciences 23 (6) (1992) 1357–1372.
[2] D.A. Bloch, B.W. Silverman, Monotone discriminant functions and their applications in rheumatology, Journal of the American Statistical Association 92 (437) (1997) 144–153.
[3] E. Boros, P.L. Hammer, J.N. Hooker, Predicting cause–effect relationships from incomplete discrete observations, SIAM Journal on Discrete Mathematics 7 (4) (1994) 531–543.
[4] E. Boros, P.L. Hammer, T. Ibaraki, K. Makino, Polynomial-time recognition of 2-monotonic positive Boolean functions given by an oracle, SIAM Journal on Computing 26 (1) (1997) 93–109.
[5] I.P. Fellegi, A.B. Sunter, A Theory for Record Linkage, Journal of the American Statistical Association 64 (1969) 1183–1210.
[6] D.N. Gainanov, On one criterion of the optimality of an algorithm for evaluating monotonic boolean functions, USSR Computational Mathematics and Mathematical Physics 24 (4) (1984) 176–181.
[7] G. Hansel, Sur le nombre des fonctions Booléennes monotones de n variables, C. R. Acad. Sc. Paris 262 (1966) 1088–1090 (in French).
[8] D.G. Horvitz, D.J. Thompson, A generalization of sampling without replacement from a finite universe, Journal of the American Statistical Association 47 (1952) 663–685.
[9] D.H. Judson, On the inference of semi-coherent structures from data, A Master's Thesis, University of Nevada, Reno, NV, USA, 1999.
[10] D.H. Judson, A partial order approach to record linkage, Federal Committee on Statistical Methodology Conference, Arlington, VA, USA, 14–16 November, 2001.
[11] A.D. Korshunov, On the number of monotone Boolean functions, Problemy Kibernetiki 38 (1981) 5–108 (in Russian).
[12] B. Kovalerchuk, E. Triantaphyllou, E. Vityaev, Monotone Boolean function learning techniques integrated with user interaction, Proceedings of the Workshop Learning from Examples vs. Programming by Demonstration, 12th International Conference on Machine Learning, Lake Tahoa, CA, 1995, pp. 41–48.
[13] B. Kovalerchuk, E. Triantaphyllou, A.S. Deshpande, Interactive learning of monotone Boolean functions, Information Sciences 94 (1996) 87–118.
[14] B. Kovalerchuk, E. Triantaphyllou, J.F. Ruiz, V.I. Torvik, E. Vitayev, The reliability issue of computer-aided breast cancer diagnosis, Computers and Biomedical Research 33 (2000) 296–313.
[15] B. Kovalerchuk, E. Vityaev, Data Mining in Finance, Kluwer Academic Publishers, Boston, MA, 2000.
[16] K. Makino, T. Ibaraki, A fast and simple algorithm for identifying 2-monotonic positive Boolean functions, in: Proceedings of ISAACS'95, Algorithms and Computation, Springer-Verlag, Berlin, Germany, 1995, pp. 291–300.
[17] I. Shmulevich, Properties and applications of monotone Boolean functions and stack filters, A Ph.D. Dissertation, Purdue University, West Lafayette, IN, USA, 1997.
[18] N.A. Sokolov, On the optimal evaluation of monotonic Boolean functions, USSR Computational Mathematics and Mathematical Physics 22 (2) (1982) 207–220.

[19] V.I. Torvik, Knowledge discovery and data mining: a guided approach based on monotone Boolean functions, Ph.D. Dissertation, Louisiana State University, Baton Rouge, LA, USA, 2002.
[20] V.I. Torvik, E. Triantaphyllou, Minimizing the average query complexity of learning monotone Boolean functions, INFORMS Journal on Computing 14 (2) (2002) 144–174.
[21] V.I. Torvik, E. Triantaphyllou, Guided inference of stochastic monotone Boolean functions, Working paper, Department of Psychiatry, University of Illinois at Chicago, Chicago, IL, USA, 2003.
[22] L.G. Valiant, A theory of the learnable, Communications of the ACM 27 (11) (1984) 1134–1142.
[23] W. Winkler, Matching and record linkage, in: B.G. Cox et al. (Eds.), Business Survey Methods, John Wiley & Sons, New York, NY, 1995.