Chapter 4 [1]

DISCOVERING RULES THAT GOVERN MONOTONE PHENOMENA

Vetle I. Torvik * and Evangelos Triantaphyllou **

*:University of Illinois at Chicago, Dept. of Psychiatry, 1601 W Taylor St, Chicago, IL 60612
Email: vtorvik@uic.edu; Web: http://arrowsmith2.psych.uic.edu/torvik

**: Louisiana State Univeristy, Dept. of Computer Science, 298 Coates Hall, Baton Rouge, LA 70803, Email: trianta@lsu.edu; Web: http://www.csc.lsu.edu/trianta

Abstract: Unlocking the mystery of natural phenomena is a universal objective in scientific research. The rules governing a phenomenon can most often be learned by observing it under a sufficiently large number of conditions that are sufficiently high in resolution. The general knowledge discovery process is not always easy or efficient, and even if knowledge is produced it may be hard to understand, interpret, validate, remember, and use. Monotonicity is a pervasive property in nature: it applies when each predictor variable has a non-negative effect on the phenomenon under study. Due to the monotonicity property, being able to observe the phenomenon under specifically selected conditions may increase the accuracy and completeness of the knowledge at a faster rate than a passive observer who may not receive the pieces relevant to the puzzle soon enough. This scenario 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. Our main goal is to pinpoint the queries that minimize the total number of queries used to completely reconstruct all of the underlying rules defined on a given finite set of observable conditions $V = \{0,1\}^n$. We summarize the optimal query selections in the simple form of selection criteria, which are near optimal and only take polynomial time (in the number of conditions) to compute. Extensive unbiased empirical results show that the proposed selection criterion approach is far superior to any of the existing methods. In fact, the average number of queries is reduced exponentially in the number of variables n and more than exponentially in the oracle's error rate.

Key Words: Monotone Boolean Functions, Active Learning, Data Mining.

---

[1] Triantaphyllou, E. and G. Felici (Eds.), **Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques**, Massive Computing Series, Springer, Heidelberg, Germany, pp. 149-192, 2005.

1. INTRODUCTION

The process of extracting new knowledge from large amounts of data is often called *Knowledge Discovery* or *Data Mining*. The general knowledge discovery process is not always easy or efficient, and even if knowledge is produced it may he hard to understand, interpret, validate, remember, and use. This chapter addresses the problem of learning monotone Boolean functions with the underlying objective to *efficiently acquire simple and intuitive knowledge that can be validated and has a general representation power.* The following key properties strengthen the argument in favor of this objective:

Key Property 1: *Monotone Boolean functions are inherently frequent in applications.*

The following three examples illustrate the versatility of the monotonicity property and how it applies to practical situations. A) Suppose a computer tends to crash when it runs a particular word processor and web browser simultaneously. Then, the computer will probably crash if it, in addition, runs other software applications. Further, suppose this computer does not tend to crash when it runs a particular CD player and web browser simultaneously. Then, it will probably not crash when it only runs the web browser (or the CD player). B) If a keyword search in a database gives interesting hits, then hits for a proper superset of these keywords is also probably going to be interesting. On the other hand, if a keyword search in a database does not give interesting hits, then hits for a proper subset of these keywords is probably not going to be interesting either. C) With all other factors constant, a student with a high Grade Point Average (GPA) is more likely to be accepted into a particular college than a student with a low GPA.

Recent literature contains a plethora of phenomena that can be modeled by using monotone Boolean functions. Such diverse phenomena include, but are not limited to, social worker's decisions, lecturer evaluation and employee selection (Ben-David, 1992), chemical carcinogenicity, tax auditing and real estate valuation (Boros *et al.*, 1994), breast cancer diagnosis and engineering reliability (Kovalerchuk *et al.*, 1996), signal processing (Shmulevich, 1997), rheumatology (Bloch and Silverman, 1997), voting rules in the social sciences (Judson, 1999), financial systems (Kovalerchuk and Vityaev, 2000b), record linkage (in administrative databases (Judson, 2001), and in bibliographic databases (Torvik *et al.*, 2004)).

Key Property 2: *Monotone Boolean functions are simple and intuitive.*

This property is perhaps the most important one when human interaction is involved since people tend to make very good use of knowledge they can easily interpret, understand, validate, and remember. Due to the increasing computational efficiency and storage capacity, the recent trend has been to increase the knowledge representation power in order to capture more complex knowledge. For example, the popular neural networks are capable of representing very complex knowledge. Unfortunately, even small neural networks can be hard to interpret and validate.

Key Property 3: *Monotone Boolean functions can represent relatively complex knowledge and still be validated.*

Validating knowledge that is generalized from a set of specific observations, which may be noisy and incomplete, is based on philosophical arguments and assumptions. Traditional statistical approaches tend to require specific modeling in small dimensions, to gain a theoretical justification for the final model. This justification is obtained at the cost of eliminating the computational feasibility of learning higher dimensional rules. On the other hand, the more general the knowledge representation is, the more one tends to lose the handle on its validation.

In practice, a great deal of time and effort is put into the knowledge discovery process. Software applications are tested, diseases are researched, database search engines are trained to be intelligent, and so on. The 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 chapter is to minimize the labor associated with gathering the data, as long as it is computationally feasible.

Monotone Boolean functions lay the ground for a simple and efficient question asking strategy, where it may be easy to pinpoint questions whose answers make incomplete knowledge more general or stochastic knowledge more accurate. Due to the underlying monotonicity property, this learning strategy may significantly increase the learning rate, as an unguided learner might not receive the relevant pieces of information early enough in the inference process. Therefore, it is highly desirable not only to be able to pose questions, but also to pose "smart" questions. The main problem addressed in this chapter is how to identify these "smart" questions in order to efficiently infer monotone Boolean functions. This chapter focuses on the case when the monotone Boolean functions are defined on the set of n-dimensional Boolean vectors $\{0,1\}^n$. This does not necessarily limit the application domain as the methodology developed in this chapter can be applied to any finite set of vectors $V \subseteq \{0,1\}^n$, and any monotone function can be represented by a set of monotone Boolean functions.

This chapter is organized as follows: The background information and the

relevant literature is reviewed in section 2. Formal definitions of the problems and their solution methodology are given in section 3. In section 4, experimental results are provided, for which a summary and discussion is given in section 5. Section 6 concludes the paper with a few final remarks.

2. BACKGROUND INFORMATION

2.1 Problem Descriptions

Let V denote a finite set of vectors defined on n variables. A vector $v \in V$ is said to *precede* another vector $w \in V$, denoted by $v \preceq w$, if and only if (iff) $v_i \leq w_i$ for $I = 1, 2, ..., n$. Here, v_i and w_i denote the I-th element of vectors v and w, respectively. Similarly, a vector $v \in V$ is said to *succeed* another vector $w \in V$, iff $v_i \geq w_i$ for $I = 1, 2, ..., n$. When v precedes (or succeeds) w, and the two vectors are distinct (i.e., $v \neq w$), then the vector v is said to *strictly precede* (or *strictly succeed*, respectively) w, denoted by $v \prec w$ (or $v \succ w$, respectively). If a vector v either precedes or succeeds w, they are said to be *related* or *comparable*. A Boolean function defined on the set of vectors $\{0,1\}^n$ is simply a mapping to $\{0,1\}$. A monotone Boolean function f is called *non-decreasing* iff $f(v) \leq f(w)$ $\forall v, w \in \{0,1\}^n : v \preceq w$, and *non-increasing* iff $f(v) \geq f(w)$ $\forall v, w \in \{0,1\}^n : v \preceq w$. This chapter focuses on non-decreasing functions, which are referred to as just *monotone*, as analogous results hold for non-increasing functions.

Monotone Boolean functions lay the ground for a simple question asking strategy, which forms the basis of this chapter. More specifically, the problem of inferring monotone Boolean functions by successive and systematic *function evaluations* (*membership queries* submitted to an oracle) is addressed. The monotone Boolean function can be thought of as a phenomenon, such as breast cancer or a computer crash, together with a set of predictor variables. The *oracle* can be thought of as an entity that knows the underlying monotone Boolean function and provides a Boolean function value in response to each membership query. In practice, it 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 inference problem is broken down by the nature of the oracle: whether it is deterministic or stochastic, and whether it is two-valued or three-valued. The simplest variant considers the guided inference of a deterministic monotone Boolean function defined on at most n Boolean variables. This case is referred to as Problem 1 which is generalized into two different problems. The first generalization includes a pair of nested monotone Boolean functions and is referred to as Problem 2. Since this problem includes two oracles, it is further broken down into three subproblems 2.1, 2.2, and 2.3, differing only in the

manner in which these two oracles are accessed. The second generalization includes stochastic membership queries and is referred to as Problem 3.

Problem 1: Inferring a Monotone Boolean Function from a Deterministic Oracle

Initially, the entire set of 2^n Boolean vectors $\{0,1\}^n$ is considered to be unclassified. That is, the values of the underlying monotone Boolean function f are all unknown and may be 0 or 1. A vector v is then selected from the set of unclassified vectors U and is submitted to an oracle as a membership query. After the vector's function value $f(v)$ is provided by the oracle, the set of unclassified vectors is reduced according to the following monotonicity constraints: $f(w) = 0, \forall w \in U: w \preceq v$, when $f(v) = 0$, or the following monotonicity constraints: $f(w) = 1, \forall w \in U: v \preceq w$, when $f(v) = 1$. Here, the relationship $v \preceq w$ holds if and only if $v_i \leq w_i$, for $I = 1, 2, ..., n$, where v_i and w_i denote the I-th Boolean elements of the vectors v and w, respectively. Vectors are then repeatedly selected from the unclassified set until they are all classified (i.e., $U = \{\}$). Given the classification of any unclassified vector, other vectors may be concurrently classified if the underlying Boolean function is assumed to be monotone. Therefore, only a subset of the 2^n vectors need to be evaluated in order to completely reconstruct the underlying function. Thus, a key problem is to select "promising" vectors so as to reduce the total number of queries (or *query complexity*).

Problem 2: Inferring a Pair of Nested Monotone Boolean Functions from Deterministic Oracle(s)

A pair of monotone Boolean functions f_1 and f_2 are called *nested* when the following relationship holds: $f_1(v) \geq f_2(v)$ (or $f_1(v) \leq f_2(v)$) $\forall v \in \{0,1\}^n$. The case when $f_1 \geq f_2$ is addressed in this chapter as analogous results hold for the case when $f_1 \leq f_2$. A single monotone Boolean function does not capture the idea of a classification intermediate to 0 and 1. However, a pair of nested monotone Boolean functions can do so. For example, some vectors might belong to a class with a high probability (i.e., where $f_1 = 1$ and $f_2 = 1$), and some might belong to the other class with a high probability (i.e., where $f_1 = 0$ and $f_2 = 0$). Other instances might not be classifiable with a satisfactorily high probability. A pair of nested monotone Boolean functions allows for this intermediate classification (i.e., where $f_1 = 1$ and $f_2 = 0$) to be incorporated. This makes the monotone Boolean function model more powerful.

Since the inference of a pair of nested monotone Boolean functions may include two oracles, it is further broken down into three subproblems 2.1, 2.2, and 2.3, differing only in the manner in which the oracle(s) are accessed. These three problems were defined to capture the main inference scenarios that may arise in real world applications.

Problem 2.1: Sequentially Inferring Nested Functions from Two Oracles

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.

Problem 2.2: Inferring Nested Functions from a 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 $f_1 \$ f_2$. 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 2.3: Inferring Nested Functions from Two Unrestricted Oracles

This problem is similar to Problem 2.1, in that two oracles are queried separately. Unlike Problem 2.1, no restrictions are put on the manner in which the two oracles are queried. At each inference step, a vector can be submitted to either 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.

Problem 3: Inferring a Monotone Boolean Function from a Stochastic Oracle

This problem is identical to Problem 1, except that the membership values are now stochastic in nature. As in Problem 1, vectors are selected from $\{0,1\}^n$ and are submitted to an oracle as membership queries. Unlike Problem 1, it is assumed that the oracle misclassifies each vector v with an unknown probability $q(v)$ 0 (0, ½). That is, for a given monotone Boolean function f, the oracle returns 1 for vector v with probability $p(v) = q(v) \times (1 - f(v)) + (1 - q(v)) \times f(v)$, and it returns 0 with probability $1 - p(v)$. It is assumed that the oracle is not misleading the inference process and is better at classifying the vectors than completely random guessing, hence the oracle's misclassification probability is assumed to be less than one half.

The stochastic inference problem involves estimating the misclassification parameter $q(v)$ for each vector v, as well as reconstructing the underlying function f. In this chapter, these two tasks are based on a maximum likelihood framework. A monotone Boolean function that is the most likely to match the underlying function, given the observed queries, is referred to as the *inferred function* and is denoted by f^*. Associated with a function f^* are the estimated misclassification probabilities which are denoted by $q^*(v)$ for each vector v.

The inference process consists of two steps that are repeated successively.

In the first step, a vector is submitted to the oracle as a query. After a vector's function value is provided by the oracle, both $q^*(v)$ and f^* may have to be updated, according to the following monotonicity property: $p(v) \# p(w)$ if and only if $v \sim w, \forall v, w \in \{0,1\}^n$. These two steps are repeated until the likelihood of the inferred function f^* matching the underlying function f is high relative to the likelihood of any of the other monotone Boolean functions matching f. In other words, the underlying function is considered completely inferred when the maximum likelihood ratio for the inferred function, denoted by $\delta(f^*)$, reaches a value that is close to 1. Again, the key problem is to select "promising" vectors so as to reduce the total number of queries required in this process.

2.2 Hierarchical Decomposition of Variables

In some applications, the variables may be monotone Boolean functions defined on a set of Boolean variables at a lower level. Kovalerchuk *et al.* (1996) decomposed five breast cancer diagnostic variables in a hierarchical manner as follows. Function $f_1(v)$ describes their "biopsy subproblem" and is defined as 1 if a biopsy is recommended for a tumor with the features described by vector v, and 0 otherwise. Function $f_2(v)$ describes their "cancer subproblem" and is defined as 1 if a tumor with the features described by v is highly suspicious for malignancy, and 0 otherwise. 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: $v_1(x_1, x_2, x_3) = x_2 \vee x_1x_3$. 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: $v_2(x_4, x_5, x_6, x_7, x_8) = x_4 \vee x_5 \vee x_6x_7x_8$. 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. Figure 1 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 $\{v_1, v_2, v_3, v_4, v_5\}$ 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.

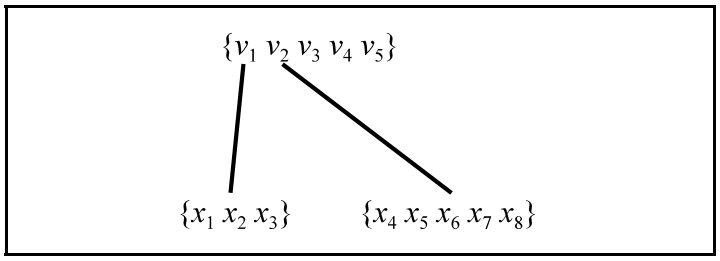


Figure 1. Hierarchical Decomposition of the Breast Cancer Diagnosis Variables.

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} = 4,096 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 4,096/104 . 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 Problem 1 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 2.1 and includes the query domain $\{0,1\}^6$.

2.3 Some Key Properties of Monotone Boolean Functions

An ordered set of related vectors $v^1 \tilde{} v^2 \tilde{} \ldots \tilde{} v^p$ is sometimes called a *chain*, while an *antichain* (or *layer*) consists of a set of mutually unrelated

vectors. When a set of vectors is partitioned into as few layers as possible, a *layer partition* is formed. Similarly, when a set of vectors is partitioned into as few chains as possible, a *chain partition* is formed. For a particular layer partition, the layers can be ordered as $L^1, L^2, ..., L^r$ so that a vector $v^i \in L^i$ cannot succeed another vector $v^j \in L^j$, if $I < j$. Let $\{0,1\}^n$ denote the set of vectors defined on n Boolean variables. The layer partition for the set $\{0,1\}^n$ is unique, while its chain partition is not unique. In fact, the way one partitions $\{0,1\}^n$ into chains can be used effectively in the inference of monotone Boolean functions. An example is the symmetric chain partition used by Hansel (1966) and Sokolov (1982) as described in section 2.4.

A *directed graph G* is often written in the form (V, E), where V denotes its set of vertices, and E denotes its set of directed edges. Here, a directed edge from vertex v to vertex w is written as (v, w). A directed graph (V, E) is called *cyclic* if it has a sequence of edges that starts and ends with a vector v: (v, v^1), (v^1, v^2), ...,$(v^r, v) \in E$. Figure 2 shows a *partially ordered set* (or *poset* for short). In general, posets can be formed by a set of vectors V together with the precedence relation $\preceq$, and are written as $(V, \preceq)$.

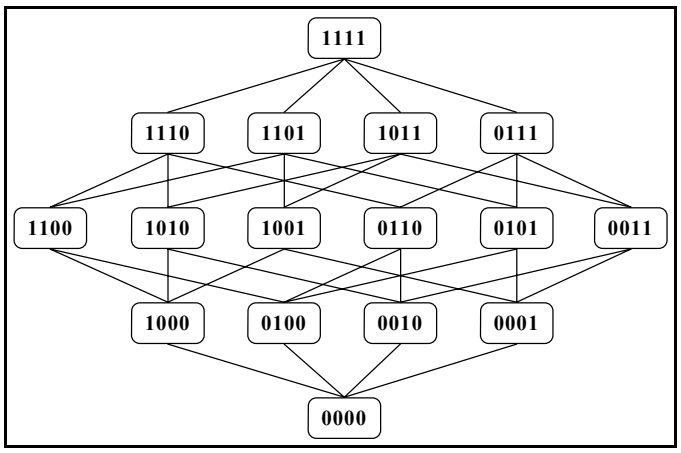


Figure 2. The Poset Formed by $\{0,1\}^4$ and the Relation $\preceq$.

A poset can be viewed as a directed graph where each vertex corresponds to a vector and each directed edge (v, w) represents the precedence relation $v \preceq w$. When drawing a poset as a directed graph, its edges' directions are often omitted without loss of information. The graph of a poset is acyclic and so all the directions can be forced upwards on a page by ordering the vertices by layers, as in Figure 2. Precedence relations that are transitively implied by other relations are considered *redundant*. For example, the precedence relation (0000) $\preceq$ (1100) is redundant because it is implied by the two precedence relations (0000) $\preceq$ (1000) and (1000) $\preceq$ (1100). For the purpose of reducing storage and simplifying the visualization of posets, redundant precedence relations are

generally omitted, as in Figure 2.

Two posets P^1 and P^2 are said to be *isomorphic* if there exists a one-to-one mapping of the vectors in P^1 to the vectors in P^2, where the precedence relations are preserved. That is, if $v^1 \leq v^2$ and $w^1 \leq w^2$, then $v^1 \sim w^1$ iff $v^2 \sim w^2$, œ $v^1, w^1 \in P^1$ and $v^2, w^2 \in P^2$. For example, the poset formed by the vectors {0000, 1001, 0100} is isomorphic to the poset formed by the vectors {1110, 1100, 1101}. Here, one possible isomorphic mapping is as follows: (0000) $\leq$ (1100), (1001) $\leq$ (1110) and (0100) $\leq$ (1101).

A vector v^* is called an *upper zero* of a Boolean function f if $f(v^*) = 0$ and $f(v) = 1$ œ $v \in \{0,1\}^n : v \supset v^*$. Similarly, a vector v^* is called a *lower unit* if $f(v^*) = 1$ and $f(v) = 0$ œ $v \in \{0,1\}^n : v \subset v^*$. 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. Boolean functions are often written in Disjunctive Normal Form (DNF) or in Conjunctive Normal Form (CNF) using the AND, OR, and NOT operations, denoted by $\vee$, $\wedge$, and ~, respectively. A DNF or a CNF representation is *minimal* if removing any of its clauses results in a different mapping $\{0,1\}^n \leq \{0,1\}$. For any monotone Boolean function f there is a one-to-one relationship between its lower units and its minimal DNF representation, as follows:

$$f(v_1, v_2, \ldots, v_n) = \bigvee_{w \in LU(f)} \left(\bigwedge_{i:w_i=1} v_i \right).$$

Similarly, there is a one-to-one relationship between the upper zeros of a monotone Boolean function f, and its minimal CNF representation as follows:

$$f(v_1, v_2, \ldots, v_n) = \bigwedge_{w \in UZ(f)} \left(\bigvee_{i:w_i=0} v_i \right).$$

For example, the monotone Boolean function defined by its lower units {110, 101} can be written in minimal DNF as $v_1v_2 \vee v_1v_3$. The corresponding upper zeros are {011, 100} and its minimal CNF representation is $v_1(v_2 \wedge v_3)$. Often the operation $\wedge$ is omitted when writing out Boolean functions, as in the previous two examples. Since the lower units and upper zeros are unique to a monotone Boolean function, so are its minimal representations in DNF and CNF. Another nice property of monotone Boolean functions is that they can be written in minimal CNF or DNF without using the NOT operation.

The set of all monotone Boolean functions defined on $\{0,1\}^n$ is denoted by M_n. For example, the set of all monotone Boolean functions defined on $\{0,1\}^2$ is given by $M_2 = \{F, v_1v_2, v_1, v_2, v_1 \vee v_2, T\}$. Here the functions T and F are defined by $f(v) = 1$, œ $v \in \{0,1\}^n$, and $f(v) = 0$, œ $v \in \{0,1\}^n$, respectively.

Let $m(f)$ denote the number of border vectors associated with a Boolean function f. It is well known (e.g., Engel, 1997) that $m(f)$ achieves its maximum value for a function that has all its border vectors on two of the most populous layers of $\{0,1\}^n$. That is, the following equation holds:

$$\max_{f \in M_n} m(f) = \binom{n}{\lfloor n/2 \rfloor} + \binom{n}{\lfloor n/2 \rfloor + 1}.$$

The borders of any monotone Boolean function f are the only vectors that require evaluations in order to completely reconstruct the function. Therefore, the value of $m(f)$ works as a lower bound on the number of queries for Problem 1.

The number of monotone Boolean functions defined on $\{0,1\}^n$ is denoted by $\Phi(n)$. That is, $\Phi(n)$ is equal to the dimension of the set M_n. All of the known values for $\Phi(n)$ are given in Table 1. For larger values of n the best known asymptotic is due to Korshunov (1981):

$$\Phi(n) \sim \begin{cases} 2^{\binom{n}{n/2}} e^{\binom{n}{n/2-1}\left(\frac{1}{2^{n/2}}+\frac{n^2}{2^{n+5}}-\frac{n}{2^{n+4}}\right)}, & \text{for even } n. \\ 2^{\binom{n}{n/2-1/2}+1} e^{\binom{n}{n/2-3/2}\left(\frac{1}{2^{(n+3)/2}}-\frac{n^2}{2^{n+6}}-\frac{n}{2^{n+3}}\right)+\binom{n}{n/2-1/2}\left(\frac{1}{2^{(n+1)/2}}+\frac{n^2}{2^{n+4}}\right)}, & \text{for odd } n. \end{cases}$$

The number of pairs of nested monotone Boolean functions defined on $\{0,1\}^n$ is simply $\Phi(n+1)$. This fact can be observed by constructing the poset 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 $f_1(v) \le f_2(v)$, $\forall v \in \{0,1\}^n$.

Table 1. History of Monotone Boolean Function Enumeration.

$\Phi(1) = 3$, $\Phi(2) = 6$, $\Phi(3) = 20$	
$\Phi(4) = 168$	by Dedekind (1897)
$\Phi(5) = 7{,}581$	by Church (1940)
$\Phi(6) = 7{,}828{,}354$	by Ward (1946)
$\Phi(7) = 2{,}414{,}682{,}040{,}998$	by Church (1965)
$\Phi(8) = 56{,}130{,}437{,}228{,}687{,}557{,}907{,}788$	by Wiedeman (1991)

2.4 Existing Approaches to Problem 1

Let $\eta(A, f)$ denote the number of queries performed by an algorithm A, when reconstructing the monotone Boolean function f. A *Teacher* can be thought of as an inference algorithm that knows the function ahead of time. It simply verifies that the function is correct by querying only the border vectors. Thus, $\eta(Teacher, f) = m(f)$, $\forall f \in M_n$. Please recall that $m(f)$ denotes the number of border vectors associated with a function f.

For any monotone Boolean function inference algorithm A, the value *of* $m(f)$ can be considered as a lower bound on the number of queries. Thus, $n(A, f) \geq m(f), \forall f \in M_n$. It turns out that it is possible to achieve fewer or the same number of queries as the upper bound on $m(f)$, for all monotone Boolean functions defined on $\{0,1\}^n$. This can be achieved by partitioning the set of vectors into chains as described in Hansel (1966). In general, there are a total of $\binom{n}{\lfloor n/2 \rfloor}$ chains in n dimensions. An inference algorithm that searches these chains in increasing length is referred to as Hansel's algorithm. A key property of the Hansel chains is that once the function values are known for all the vectors in all the chains of length k, the function values are unknown for at most two vectors in each chain of the next length k+2. Proof of this property can be found in both Hansel (1966) and Sokolov (1982). As a result, Hansel's algorithm results in fewer or the same number of queries as the upper bound on $m(f)$ as follows. When n is odd, the shortest chains contain two vectors each, and there are a total of $\binom{n}{\lfloor n/2 \rfloor}$ chains. In this case, the maximum number of queries used by Hansel's algorithm is $2\binom{n}{\lfloor n/2 \rfloor} = \binom{n}{\lfloor n/2 \rfloor} + \binom{n}{\lfloor n/2 \rfloor + 1}$. Similarly, when n is even, there are $\binom{n}{n/2} - \binom{n}{n/2+1}$ chains of length one, and $\binom{n}{n/2+1}$ chains of length greater than one. In this case, the maximum number of queries is $\binom{n}{n/2} - \binom{n}{n/2+1} + 2\binom{n}{n/2+1} = \binom{n}{n/2} + \binom{n}{n/2+1}$. That is, the following inequality holds:

$$\varphi(\textit{Hansel}, f) \leq \max_{g \in M_n} m(g) = \binom{n}{\lfloor n/2 \rfloor} + \binom{n}{\lfloor n/2 \rfloor + 1}, \forall f \in M_n.$$

The algorithm described in Sokolov (1982) is also based on the Hansel chains. In contrast to Hansel's algorithm, it considers the chains in the reverse order (i.e., in decreasing length) and performs the binary search within each chain. It turns out that Sokolov's algorithm is much more efficient for functions that have all their border vectors in the longer Hansel chains. As an example, consider the monotone Boolean function T. This function has only one border vector (00...0), which is located in the longest chain. For this function, Sokolov's algorithm performs at most $\lfloor log_2(n) \rfloor + 1$ evaluations, while Hansel's algorithm needs at least $\binom{n}{\lfloor n/2 \rfloor}$ evaluations. For instance, when n = 20 this translates into at least 184,756 evaluations performed by Hansel's algorithm and at most 5 evaluations performed by Sokolov's algorithm.

Sokolov's algorithm does not satisfy the upper bound, as the following example shows. Suppose that $n > 4$ and even, and the monotone Boolean function to be inferred is defined by $f(v) = 1 \Leftrightarrow v \in \{0,1\}^n : |v| \geq n/2$, and 0 otherwise. Then the set of border vectors is $\{v, |v| = n/2 \text{ or } n/2\text{-}1\}$ and $m(f) = \binom{n}{\lfloor n/2 \rfloor} + \binom{n}{\lfloor n/2 \rfloor + 1}$. In Sokolov's algorithm, the first vector w^1 submitted for evaluation is a border vector since $|w^1| = n/2$. The second vector w^2 is not a border vector because $|w^2| = \lfloor 3n/4 \rfloor \ldots n/2$ and $n/2$-1. Therefore, the following inequality holds:

$$n(\textit{Sokolov}, f) > \binom{n}{\lfloor n/2 \rfloor} + \binom{n}{\lfloor n/2 \rfloor + 1}, \text{ for at least one } f \in M_n.$$

In an attempt to provide a unified efficiency testing platform, Gainanov (1984) proposed to compare inference algorithms based on the number of evaluations needed for each border vector. To that end, he presented an algorithm that searches for border vectors one at a time, and we refer to this algorithm as FIND-BORDER. At the core of the algorithm is a subroutine that takes as input any unclassified vector v, and finds a border vector by successively evaluating adjacent vectors. This subroutine is also used in the algorithms of Boros *et al.* (1997), Makino and Ibaraki (1995), and Valiant (1984). As a result, any inference algorithm A that feeds unclassified vectors to this subroutine satisfies the following upper bound:

$$n(A, f) \# m(f)(n+1), \text{ œ} f \text{ O } M_n.$$

For the majority of monotone Boolean functions, the expression $m(f)(n+1)$ is greater than or equal to 2^n, in which cases the bound is trivial.

Earlier work on monotone Boolean function inference (such as Hansel, 1966; Sokolov, 1982; Gainanov, 1984) focuses on reducing the query complexity. More recent work (like Boros *et al.*, 1997; Makino and Ibaraki, 1997; Fredman and Khachiyan, 1996) considers both the query complexity and the computational complexity. The problem of inferring a monotone Boolean function via membership queries is equivalent to many other computational problems in a variety of fields (see, for instance, Bioch and Ibaraki, 1995; Eiter and Gottlob, 1995). In these applications, algorithms that are efficient in terms of query and computational complexity are used.

In practice, queries often involve some sort of effort, such as consulting with experts, performing experiments or running simulations. For such applications, queries far surpass computations in terms of cost. Therefore, this chapter focuses on minimizing the query complexity as long as it is computationally feasible.

2.5 An Existing Approach to Problem 2

Kovalerchuk *et al.* (1996) considered the problem of inferring a pair of nested monotone Boolean functions. Their algorithm, which exhibited a promising efficiency in their cancer diagnosis application, is an extension of Hansel's inference algorithm for a single monotone Boolean function. However, the algorithm performance analysis is far from conclusive as a single application represents a single pair of nested monotone Boolean functions.

2.6 Existing Approaches to Problem 3

The problem of guided inference in the presence of stochastic errors is referred to as sequential design of experiments in the statistics community. The field of optimal experiment design (Federov, 1972) contains various optimality criteria that are applicable in a sequential setting. The most common vector selection criterion is based on instantaneous variance reduction. Other selection criteria, such as the maximum information gain used in MacKay (1992), and Tatsuoka and Ferguson (1999), have been studied. However, no guided inference studies using a maximum likelihood framework were found in the literature.

The theory of optimal experiment design is the most extensive for simple regression models (Federov, 1972). Fortunately, efficient guided inference for more complex models have been studied, such as the feed forward neural networks in Cohn (1996), even though a sound theory has not been established. In fact, the same article reported a convergence problem for which a partial remedy was introduced in Cohn (1995).

2.7 Stochastic Models for Problem 3

Suppose a set of observed vectors $V = \{v^1, v^2, ..., v^k\}$ is given. For a given number of queries m, let $m_z(v)$ be the number of times the oracle classified vector v as z (for $z = 0$ and 1, and $v \in V$). Associated with a monotone Boolean function f, the number of errors it performs on the set of observations is given by:

$$e(f) = \sum_{i=1}^{k} (f(v^i)m_0(v^i) + (1-f(v^i))m_1(v^i)).$$

It is assumed that the oracle misclassifies each vector v with a probability $q(v) \in (0, ½)$. That is, for a given monotone Boolean function f, the oracle returns for vector v :

1 with probability $p(v) = q(v)\times(1 - f(v)) + (1 - q(v))\times f(v)$, and
0 with probability $1- p(v)$.

A key assumption is that the misclassification probabilities are all less than ½, otherwise it would not be possible to infer the correct monotone Boolean function. If the sampled values are considered fixed, their joint probability distribution function can be thought of as the likelihood of function f matching the underlying function as follows:

$$L(f) = q^{e(f)}(1 - q)^{m - e(f)}.$$

The likelihood value of a particular monotone Boolean function decreases exponentially as more observations are added and therefore this value is generally very small. However, the likelihood ratio given by:

$$\lambda(f^*) = \frac{L(f^*)}{\sum_{f \in F(V)} L(f)}$$

measures the likelihood of a particular function f^* relative to the likelihood of all possible monotone Boolean functions $F(V)$, defined on the set of vectors V. Note that when the set of vectors V is equal to $\{0,1\}^n$, then the set of all possible monotone Boolean functions $F(V)$ is equal to M_n.

The goal of the maximum likelihood problem is to find a monotone Boolean function $f^* \in F(V)$, so that $L(f^*) \geq L(f) \; \forall f \in F(V)$. Assuming that the misclassification probabilities $q(v)$ are all less than ½, this problem is equivalent to identifying a monotone Boolean function f^* that minimizes the number of errors $e(f^*)$ (Boros *et al.*, 1995). Note that if q can take on values greater than ½, then the maximum likelihood solution may maximize the number of errors, as demonstrated by Boros *et al.* (1995). In this chapter, error maximization is avoided by restricting q to be less than ½; existence of such a solution is shown in Torvik and Triantaphyllou (2004).

The error minimization problem can be converted into an integer maximization problem as follows:

$$\min \sum_{i=1}^{k} (f(v^i)m_0(v^i) + (1-f(v^i))m_1(v^i)) =$$

$$\min e(f) =$$

$$\min(-\sum_{i=1}^{k} (f(v^i)(m_1(v^i) - m_0(v^i)) + \sum_{i=1}^{k} m_1(v^i)).$$

Since the term $\sum_{i=1}^{k} m_1(v^i)$ is constant, it can be removed from the optimization objective. Furthermore, maximizing a particular objective function is equivalent to minimizing the negative of that objective function, resulting in the following simplified integer optimization problem:

$$\min \sum_{i=1}^{k} (p(v^i) - n(v^i))(m_1(v^i) + m_0(v^i))$$

subject to $f(v^i) \# f(v^j) \; \forall v^i, v^j \in V: v^i \sim v^j$, and
$f(v^i) = 0$ or 1.

This problem is known as a maximum closure problem, which can be converted into a maximum flow problem (Picard, 1976). The most efficient algorithms developed for the maximum flow problem use the idea of preflows developed by Karzanov (1974). For example, the lift-to-front algorithm (e.g., Cormen *et al.*, 1997) takes $O(V^3)$ time. The fact that this problem can be solved in polynomial time is a nice property of the single q parameter model. For two dimensional problems (i.e., $V \subset \mathbb{R}^2$), the minimum number of errors can also be guaranteed via a dynamic programming approach (Boros and Silverman, 1997).

A more complex error model can potentially maintain as many parameters

$$\max \sum_{i=1}^{k} f(v^i)(m_1(v^i) - m_0(v^i))$$

as the size of the domain V. That is, each vector v may have an associated unique parameter $p(v)$. In this case, minimizing the weighted least squares:

$$\text{subject to } p(v^i) \# p(v^j) \text{ œ } v^i, v^j \text{ O } V: v^i \tilde{} v^j,$$

where

$$\bar{p}(v^i) = \frac{m_1(v^i)}{m_1(v^i) + m_0(v^i)}, \text{ for } i = 1, 2, ..., k,$$

yields a maximum likelihood solution (Robertson *et al.*, 1988). This is a hard optimization problem, and several algorithms have been developed to solve it optimally and near optimally. The Pooled Adjacent Violators Algorithm (PAVA) by Ayer *et al.* (1955) only guarantees optimality when (V, $\tilde{}$) forms a chain poset (also referred to as a simple order). The Min-Max algorithm developed by Lee (1983) and the Isotonic Block Class with Stratification (IBCS) algorithm by Block *et al.* (1994) guarantee optimality for the general poset but both algorithms can potentially consume exponential time. Unfortunately, no polynomial algorithm for the general poset was found in the literature.

In addition to the full parametric model, there are models of intermediate parametric complexity. One example is the logistic regression model with non-negativity constraints on its parameters, as used for record linkage in databases by Judson (2001). A monotone decision tree approach can be found in Makino *et al.* (1999), and a sequential monotone rule induction approach can be found in Ben-David (1992 and 1995).

It should be noted that the single parameter error model considered in this chapter is somewhat restrictive, in the sense that it does not estimate misclassification probabilities that vary across the vectors. However, the goal of this chapter is to efficiently uncover the underlying monotone Boolean function and not necessarily come up with accurate estimates for the individual errors. The fixed misclassification probability assumption does not affect the capability of the inference methodology as will be demonstrated in the subsequent sections. The assumption is simply used to estimate the error rate and the confidence in having inferred the correct function, and a more accurate estimate of the maximum likelihood ratio may require a substantial increase in computational complexity, as for the full parametric model described above.

3. INFERENCE OBJECTIVES AND METHODOLOGY

3.1 The Inference Objective for Problem 1

An inference algorithm that performs fewer queries than another algorithm when reconstructing a particular deterministic monotone Boolean function is considered more efficient on that particular function. However, it has not been clear how to compare algorithms on the entire class of monotone Boolean functions defined on $\{0,1\}^n$.

The main existing algorithms by Hansel (1966), Sokolov (1982), and Gainanov (1984) focus on the upper bounds of their query complexities. Unfortunately, the worst case scenario reflects the algorithm performance on a few specific functions. It does not reflect what to expect when executing the algorithm on an arbitrary monotone Boolean function. For example, algorithms that implement Gainanov's subroutine (which we refer to as FIND-BORDER) indirectly suggest minimizing the upper bound on the number of evaluations per border vector. These algorithms greatly favor the simplest functions (which may only have a single border vector) over the complex functions (with up to $\binom{n}{\lfloor n/2 \rfloor} + \binom{n}{\lfloor n/2 \rfloor + 1}$ border vectors). Kovalerchuk *et al.* (1996) demonstrated promising results for a Hansel based inference algorithm on a real world application. However, their performance analysis is far from conclusive as a single application represents a single pair of monotone Boolean functions.

With no prior knowledge (other than monotonicity) about the inference application, each function is equally likely to be encountered and should therefore carry the same weight in the objective. The objective for this problem is to develop an algorithm that minimizes the average number of queries over the entire class of monotone Boolean functions defined on the set $\{0,1\}^n$. This objective can be expressed mathematically as follows:

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

The objective $Q(n)$ represents the entire class of monotone Boolean functions M_n. As such, it provides a better indication of what to expect when executing an algorithm on an arbitrary monotone Boolean function.

3.2 The Inference Objective for Problem 2

The approach taken to this problem is analogous to that of Problem 1. The minimum average number of queries for Problem 2.k (for k = 1, 2, and 3) can be expressed mathematically as follows:

$$Q_k(n) = \min_{A_k} \frac{\sum_{f_1, f_2 \in M_n;\, f_2 \le f_1} \varphi(A_k, f_1, f_2)}{\Psi(n+1)}.$$

Here, $\eta(A_k, f_1, f_2)$ denotes the number of queries performed by algorithm A_k, in reconstructing the pair of nested monotone Boolean functions f_1 and f_2

defined on the set $\{0,1\}^n$. Here, A_1, A_2, and A_3 denote algorithms designed for Problems 2.1, 2.2, and 2.3, respectively. Please recall from section 2 that the number of pairs of nested monotone Boolean functions defined on the set $\{0,1\}^n$ is equal to $Q(n+1)$, the number of monotone Boolean functions defined on the set $\{0,1\}^{n+1}$.

Since these 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 2.1 and 2.3, where two oracles are accessed individually. For Problem 2.2, 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 query to the three-valued oracle will be counted as two queries.

3.3 The Inference Objective for Problem 3

The approach taken to Problem 3 is similar to that of Problems 1 and 2. The goal is to minimize the average number of queries needed to completely reconstruct the underlying monotone Boolean function, expressed mathematically as follows:

$$\min_{A} \frac{\sum_{f \in M_n} \varphi(A, f, q)}{\Psi(n)}.$$

Here, $\Theta(A, f, q)$ denotes the expected number of queries performed by algorithm A in completely reconstructing the underlying monotone Boolean function f from an oracle with a fixed misclassification probability q. Completely reconstructing the underlying function translates into making the likelihood ratio $\Omega(f^*)$ for the inferred function f^* reach a sufficiently high value (e.g., 0.99).

It should be stressed that the misclassification probability q is unknown and ranges from 0 up to ½. However, it is expected that the average number of queries will increase significantly with q, since, by definition, it approaches infinity as q approaches ½, and it is finite when q is equal to 0. Therefore, the average over a large range q may not be an accurate prediction of how many queries to expect for a particular application. The average query complexity will therefore be evaluated as a function of n and q, even though q is unknown.

3.4 Incremental Updates for the Fixed Misclassification Probability Model

Suppose the error minimizing function f_{old}^* and its misclassification parameter q_{old}^*, associated with a set of vectors $V = \{v^1, v^2, ..., v^k\}$ and their $m_0(v)$ and $m_1(v)$ values, are given. When a new vector is classified by the oracle

(i.e., $m_z(v)$ 7 $m_z(v) + 1$), the function f_{old}* and its misclassification parameter q_{old}* may have to be updated. Since the new error minimizing function is likely to be close to the old function, it may be inefficient to solve the entire problem over again.

Simply stated the incremental problem consists of finding f_{new}* and consequently q_{new}* when $m_z(v)$ 7 $m_z(v) + 1$. If the new classification is consistent with the old function (i.e., $f_{old}*(v) = z$), then the old function remains error minimizing (i.e., $f_{old}* = f_{new}*$). Therefore, the number of errors remains the same and the misclassification estimate is reduced to $q_{new}* = e(f_{old}*)/(m_{old} + 1)$. Note that this case is the most likely one since it occurs with an estimated probability of 1- q_{old}* $ ½. If, on the other hand, the new classification is inconsistent with the old function (i.e., $f_{old}*(v) = 1- z$), the old function may or may not remain error minimizing. The only case in which the old function does not remain error minimizing is when there is an alternative error minimizing function f_a* on the old data for which $f_a*(v) = z$. In this case f_a* is error minimizing for the new data.

The number of possible error minimizing functions may be exponential in the size of the set V, and therefore storing all of them may not be an efficient solution to this problem. To avoid this computational burden an incremental algorithm such as the one described in Torvik and Triantaphyllou (2004) can be used.

3.5 Selection Criteria for Problem 1

When computing the optimal solutions, many different and complex posets are encountered. The optimal vectors of these posets seemed to display two general properties (Torvik and Triantaphyllou, 2002). First, the optimal vectors tend to be in the *vertical middle*. More specifically, all posets observed in the inference process when n is 4 or less have at least one optimal vector in the most populous (middle) layer. This observation alone is not sufficient to pinpoint an optimal vector. The second property observed is that the optimal vectors also tend to be *horizontal end points*.

Now consider creating a selection criterion based on the ideas of vertical middle and horizontal end points. Suppose a subset of unclassified vectors, $V = \{v^1, v^2, ..., v^p\}$ is given. Let $K_1(v^i)$ and $K_0(v^i)$ be the number of vectors that are concurrently classified when $f(v^i)$ equals to 1 and 0, respectively. Invariably selecting a vector v with the minimum $|K_1(v) - K_0(v)|$ value guarantees the minimum average number of queries for inference problems with n strictly less than 5 (Torvik and Triantaphyllou, 2002).

Unfortunately, this selection criterion is not optimal for all the posets generated for n equal to 4. It is only optimal for the subset of posets encountered when using the criterion $min|K_1 - K_0|$. Another drawback is that it is not optimal for the inference problem when n is equal to 5. However, the

criterion is probably close to optimal since the larger posets eventually decompose into smaller posets.

It is important to note that what may look like intuitive criteria (without the consultation of optimal solutions) may lead to poor performance and ambiguous choices. For example, it may seem reasonable to attempt to classify as many vectors as possible for each query. The two criteria $max(K_1(v) + K_0(v))$ and $max(K_1(v)K_0(v))$ are consistent with this philosophy (see Judson, 1999). However, they are extremely counterproductive to minimizing the average query complexity and should be avoided. As an example, consider the set of vectors $\{0,1\}^4$. The criterion $max(K_1(v) + K_0(v))$ selects either the (0000) or the (1111) vector, which happens to maximize the average number of queries. The criterion $max(K_1(v)K_0(v))$ ties the entire set of vectors, and is therefore the choice of vector is ambiguous.

There is a logical explanation for why these two selection criteria are counterproductive. Vectors that are able to concurrently classify more vectors are also more likely to be classified by others. Following this line of thought, the selection criterion $min(K_1(v) + K_0(v))$ seems reasonable. This criterion is similar to $min|K_1(v) - K_0(v)|$, but it does not satisfy the same optimality conditions for the inference problem when n is equal to 4.

3.6 Selection Criteria for Problems 2.1, 2.2, and 2.3

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 $Q(n+1)$. That is, $Q_3(n) = Q(n+1)$. 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: $\{0,1\}^n \times \{f_2, f_1\}$. Each of the vertices in the resulting poset $(\{0,1\}^{n+1}, \tilde{})$, 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 $\{0,1\}^{n+1}$.

The selection criterion $min|K_1(v)-K_0(v)|$ was shown to be very efficient in minimizing the average number of queries in Problem 1. It will therefore be used for the three nested problems with a slight modification. The query domain for the nested case is made up of the set of vectors $\{0,1\}^n \times \{f_2, f_1\}$. For a vertex labeled ($v$ f_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 access to the oracles is unrestricted (i.e., Problem 2.3), 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 2.3.

For sequential oracles (i.e., Problem 2.1), 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 2.2), 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.

3.7 Selection Criterion for Problem 3

The status of the inference process will be considered to be in one of three stages. Stage 1 starts with the first question and lasts until a deterministic monotone Boolean function is obtained. During Stage 1 only vectors that may take on both 0 and 1 values are queried. As a result, no (identifiable) errors are observed in Stage 1, and thus the monotone Boolean function inferred during Stage 1 is deterministic. This function, however, may or may not be the correct function. In fact, the probability that it is the correct function is equal to the probability that no misclassifications were made: $(1- q)^m$, where m is the number of questions used during Stage 1 and q is the true misclassification probability. This probability decreases rapidly with m, regardless of the value of q. Therefore, the queries performed after Stage 1 will benefit greatly from a reduction in the number of Stage 1 queries. Please note that since no inconsistencies have been observed, there is no way to properly estimate q at this point.

After a deterministic monotone Boolean function is obtained in Stage 1, the inference process enters Stage 2. At this point it is unclear as to how to select queries for Stage 2, so a random selection procedure will be used for this stage. After the first error occurs in Stage 2, the inference process enters Stage 3, in which it will remain until termination. Stage 3 is the focus of this chapter, because it is the only stage in which the likelihood ratio can be properly evaluated and q can be estimated based on the observed vectors.

Please recall that the likelihood function is given by:

$$L(f) = q^{e(f)}(1 - q)^{m - e(f)},$$

and the likelihood ratio is given by:

$$\lambda(f^*) = \frac{L(f^*)}{\sum_{f \in F(V)} L(f)}.$$

As an example of the likelihood ratio computations consider the example data given in Table 2. The likelihood values for the all the possible monotone Boolean functions are given in Table 3. The function $f^* = v_1v_3 \vee v_2v_3$ produces 16 errors. Its associated estimated misclassification probability q^* is 16/36 = 4/9, since the total number of observations is $m = 36$. Therefore, the likelihood value of this function $L(f^*)$ is $(4/9)^{16}(1 - 4/9)^{36-16} = 1.818\times10^{-11}$. Notice how small this value is after only 36 observations. Adding up the likelihood values the monotone Boolean functions yields $(13\times1.455 + 2\times1.536 + 5\times1.818)\times10^{-11} = 3.107\times10^{-10}$. Then the maximum likelihood ratio is computed as follows: $\delta(f^*) = 1.818\times10^{-11}/3.107\times10^{-10} = 0.0585$.

Table 2. A Sample Data Set for Problem 3.

v	$m_1(v)$	$m_0(v)$	$m_1(v) - m_0(v)$
111	0	1	-1
110	3	5	-2
101	4	1	3
11	3	1	2
100	4	5	-1
10	2	0	2
1	3	3	0
0	1	0	1

Table 3. Example Likelihood Values for All Functions in M_3.

f	$e(f)$	$q(f)$	$L(f)$	$\delta(f)$
F	20	½	1.455×10^{-11}	0.0468
$v_1v_2v_3$	21	½	1.455×10^{-11}	0.0468
v_1v_2	23	½	1.455×10^{-11}	0.0468
v_1v_3	18	½	1.455×10^{-11}	0.0468
$v_1v_2 \vee v_1v_3$	20	½	1.455×10^{-11}	0.0468
v_1	21	½	1.455×10^{-11}	0.0468
v_2v_3	19	½	1.455×10^{-11}	0.0468
$v_1 \vee v_2v_3$	19	½	1.455×10^{-11}	0.0468
$v_1v_3 \vee v_2v_3$	16	36989	1.818×10^{-11}	0.0585
$v_1v_2 \vee v_1v_3 \vee v_2v_3$	18	½	1.455×10^{-11}	0.0468
$v_1v_2 \vee v_2v_3$	21	½	1.455×10^{-11}	0.0468
v_2	19	½	1.455×10^{-11}	0.0468

$v_1 \vee v_2$	17	17/36	1.536×10^{-11}	0.0495
$v_2 \vee v_1v_3$	16	36989	1.818×10^{-11}	0.0585
v_3	16	36989	1.818×10^{-11}	0.0585
$v_2 \vee v_3$	16	36989	1.818×10^{-11}	0.0585
$v_1 \vee v_2 \vee v_3$	17	17/36	1.536×10^{-11}	0.0495
$v_1 \vee v_3$	19	½	1.455×10^{-11}	0.0468
$v_3 \vee v_1v_2$	18	½	1.455×10^{-11}	0.0468
T	16	36989	1.818×10^{-11}	0.0585

Now let us return to the vector selection (or guided inference) problem. As shown above, the probability that the correct function is inferred during Stage 1 decreases rapidly with the number of queries used during that stage. Therefore, the selection criterion $min|K_0(v) - K_1(v)|$ will be used as a standard for Stage 1, when comparing different approaches for the following Stage 3. This avoids bias in the sense that all Stage 3 approaches will benefit from using $min|K_0(v) - K_1(v)|$ during Stage 1.

One important property of the selection criterion for Stage 3 is that the maximum likelihood ratio converges to 1. It is possible to define selection criteria that do not converge. If, for example, the same vector is invariably selected, the estimated value of q will converge to its true value. In this case, the likelihood values may remain equal for several monotone Boolean functions and hence the maximum likelihood ratio will never converge to 1.

Intuition may lead to an inefficient selection criterion. For example, let $E_z(v)$ be defined by the number of errors associated with assigning the function value $f(v)$ to z, as follows:

$$E_0(v) = \sum_{w \le v} m_1(w) - m_0(w), E_1(v) = \sum_{v \le w} m_0(w) - m_1(w).$$

Then, consider defining the vector v which "contributes the most errors" by $max(E_0(v)+E_1(v))$. This vector selection criterion may lead to the same vector being invariably queried and hence it might suffer from convergence problems, as will be demonstrated empirically in section 4.

The likelihood framework seems to form a great basis for defining a Stage 3 vector selection criterion. Since the goal is to make the likelihood ratio converge to 1 as fast as possible, a reasonable approach would be to select the vector that maximizes the expected maximum likelihood ratio at each inference step. To do this, the expected maximum likelihood ratio $\Theta(v) = p(v)\Theta_1(v) + (1 - p(v))\Theta_0(v)$ has to be estimated for each vector v. Here $\Theta_z(v)$ denotes the resulting maximum likelihood ratio when $f(v) = z$ is observed. Please recall that $p(v)$ is the probability of observing $f(v) = 1$. That is, it can be estimated by $p^*(v) = q^*(1- f^*(v)) + (1 - q^*)f^*(v)$.

As an example consider observing the vector (001). Table 4 gives the updated likelihood ratios for each monotone Boolean function when $m_z(001) = m_z(001) + 1$, for $z = 0$ and 1. For a monotone Boolean function f, and a classification z, $e_z(001, f)$ and $\lambda_z(001, f)$ here denote the updated number of errors and the likelihood ratio, respectively. The updated maximum likelihood ratios are $\lambda_1(001) = \lambda_1(001, T) = 0.0649$ and $\lambda_0(001) = \lambda_0(001, v_1v_3 \vee v_2v_3) = 0.0657$. Since the optimal function assigns the vector (001) to 0 (i.e., $f^*(001) = 0$), the estimated probability of observing $f(001) = 1$ is given by $p^*(001) = q^* = 4/9$. Therefore, the expected maximum likelihood ratio when querying vector 001 is given by $\Delta\lambda(001) = p^*(001)\lambda_1(001) + (1- p^*(001))\lambda_0(001) = 4/9 \times 0.0649 + 5/9 \times 0.0657 = 0.0653$.

Similar computations for the other vectors yield $\Delta\lambda(000) = 0.0651$, $\Delta\lambda(010) = 0.0654$, $\Delta\lambda(011) = 0.0592$, $\Delta\lambda(100) = 0.0652$, $\Delta\lambda(101) = 0.0592$, $\Delta\lambda(110) = 0.0654$, and finally $\Delta\lambda(111) = 0.0592$. The vectors with the largest expected likelihood ratio value are (010) and (110). Since no further improvements of the selection criterion is immediately obvious, ties are broken arbitrarily.

The simulations in section 4 reveal the efficiency of the selection criterion $max\ \Delta\lambda(v)$ in terms of the query complexity. In terms of computational complexity it may take an exponential time (in the size of V) to compute $max\ \Delta\lambda(v)$. Since the computational time for incrementally finding the inferred function is of $O(V^2)$, it would be nice to find a selection criterion that does not take more time than this and still makes the likelihood converge to 1 at a faster rate than randomly selecting vectors.

Table 4. Updated Likelihood Ratios for $m_z(001) = m_z(001) + 1$.

f	$\lambda(f)$	$e_1(001, f)$	$\lambda_1(001, f)$	$e_0(001, f)$	$\lambda_0(001, f)$
F	0.0468	21	0.0462	20	0.0468
$v_1v_2v_3$	0.0468	22	0.0462	21	0.0468
v_1v_2	0.0468	24	0.0462	23	0.0468
v_1v_3	0.0468	19	0.0462	18	0.0474
$v_1v_2 \vee v_1v_3$	0.0468	21	0.0462	20	0.0468
v_1	0.0468	22	0.0462	21	0.0468
v_2v_3	0.0468	20	0.0462	19	0.0468
$v_1 \vee v_2v_3$	0.0468	20	0.0462	19	0.0468
$v_1v_3 \vee v_2v_3$	0.0585	17	0.0522	16	0.0657
$v_1v_2 \vee v_1v_3 \vee v_2v_3$	0.0468	19	0.0462	18	0.0474
$v_1v_2 \vee v_2v_3$	0.0468	22	0.0462	21	0.0468
v_2	0.0468	20	0.0462	19	0.0468
$v_1 \vee v_2$	0.0495	18	0.0469	17	0.0529
$v_2 \vee v_1v_3$	0.0585	17	0.0522	16	0.0657

v_3	0.0585	16	0.0649	17	0.0529
$v_2 \vee v_3$	0.0585	16	0.0649	17	0.0529
$v_1 \vee v_2 \vee v_3$	0.0495	17	0.0522	18	0.0474
$v_1 \vee v_3$	0.0468	19	0.0462	20	0.0468
$v_3 \vee v_1 v_2$	0.0468	18	0.0469	19	0.0468
T	0.0585	16	0.0649	17	0.0529

One such possibility may be based on the inferred border vectors. For the sake of argument suppose that the underlying monotone Boolean function f to be inferred is known. Then randomly selecting vectors from its corresponding border vectors will make the maximum likelihood ratio converge to 1. As the number of queries m goes to infinity, the ratios $m_0(v)/(m_0(v) + m_1(v)) \; \forall \; v \in \mathrm{LU}(f)$ and $m_1(w)/(m_0(w) + m_1(w)) \; \forall \; w \in \mathrm{UZ}(f)$ all converge to q. The number of errors performed by any other monotone Boolean function g is at least $x = min\{min\{m_1(v) - m_0(v), v \in \mathrm{LU}(f)\}, min\{m_0(w) - m_1(w), w \in \mathrm{UZ}(f)\}\}$ greater than the number of errors performed by function f. Furthermore, $x \geq qm - (1-q)m = m(2q - 1)$ for large m. That is, the number of additional errors increases at least linearly with m. Then, as m goes to infinity, so does the number of additional errors performed by each of the other monotone Boolean functions. That is, the relative likelihoods $L(f)/L(g) > (q/(1-q))^x$ converge to 0 as m goes to infinity. Since the number of other monotone Boolean functions is a finite number that does not depend on m, the likelihood ratio $\delta(f) = \mathrm{L}(f) / (\mathrm{L}(f) + \Sigma L(g))$ converges to 1 as m goes to infinity.

Focusing the queries at the border vectors of the underlying function probably allows this convergence to occur at a faster rate than randomly selecting from all the vectors. In situations where the underlying function is unknown, it may be that focusing the queries on the border vectors of the inferred function (i.e., $v \in \mathrm{LU}(f^*) \cup \mathrm{UZ}(f^*)$) is better than completely random selection. In the long run, an inferred border vector will not prevail if it is not an underlying border vector. Since the misclassification rate is less than ½, the rate at which the incorrectly classified inferred border vectors become correctly classified is greater than the rate at which correctly classified inferred border vectors become incorrectly classified. Therefore, in the long run all the classifications become correct when the queries are selected from the set of border vectors of the inferred function.

Notice that this convergence holds even if the misclassification probability is different for each vector, as long as they are all less than ½. Another added benefit is that finding the border vectors is easy, since they are readily available from the inferred function f^*. In fact, a simple modification of the incremental maximum flow algorithm can store each of these vectors as they are found. For each monotone Boolean function there are at most $O(V)$ border vectors in a set of vectors V. During the inference process the inferred function may take on any of these monotone Boolean functions. Therefore, randomly selecting one

of the border vectors takes O(V) time.

4. EXPERIMENTAL RESULTS

4.1 Experimental Results for Problem 1

The preexisting inference algorithms described in section 2 do not specify which vector to select when there are ties. In particular, the Sokolov and Hansel algorithms may have to choose between two vectors that make up the middle of a particular chain. Furthermore, the subroutine FIND-BORDER needs to be fed unclassified vectors, of which there may be many. Even the selection criterion $min|K_1\text{-}K_0|$ may result in 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 Figure 3 are based on an exhaustive analysis (i.e., all the monotone functions were generated) for n up to and including 5. Random samples of 2,000 functions were generated for n = 6, 7, and 8; while for n = 9, 10, and 11 they were composed of 200 functions; the functions were generated using the algorithm described in Torvik and Triantaphyllou (2002).

The Horvitz-Thompson (1952) estimator is used to compute the averages for n greater than 5. The average number of queries is normalized by the maximum possible number of queries 2^n so that the magnitudes of the averages in Figure 3 are not overshadowed by the large values obtained for n equal to 11. 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 in Figure 3 and the horizontal line *Average Number of Queries* / 2^n = 1 (not shown in the figure) can be thought of as the benefit of the monotone assumption. This is due to the fact that 2^n is the number of required queries when the underlying function is not necessarily monotone.

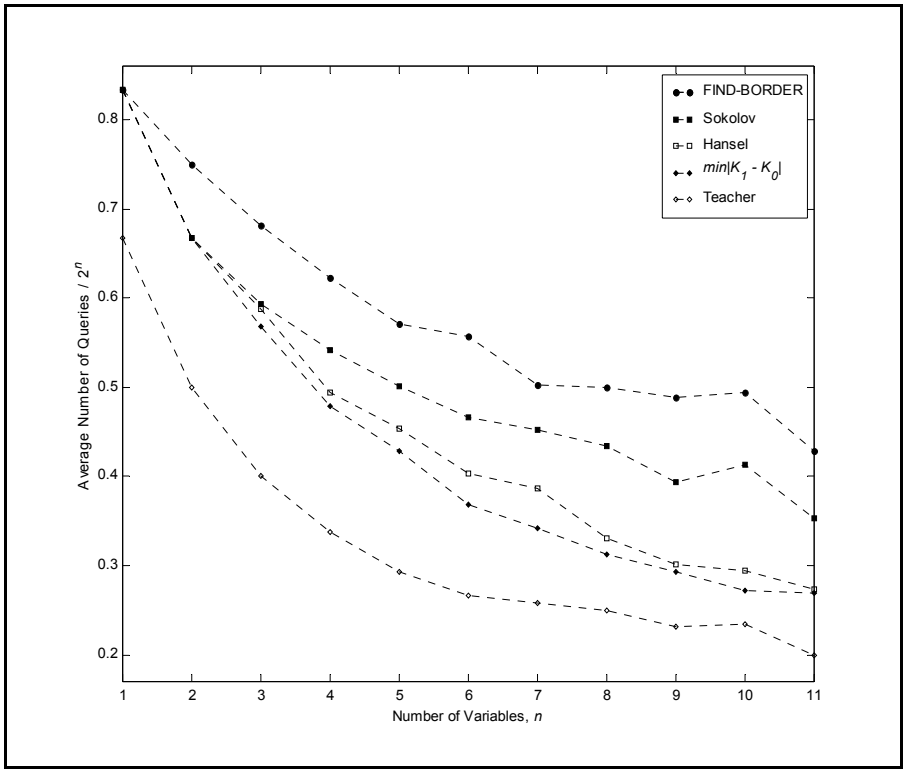


Figure 3. The Average Query Complexities for Problem 1.

The curve titled "Teacher" represents the lower bound on the number of queries for every single function. Therefore, it is expected that a few extra queries are required on the average. Since the heuristic based on the selection criterion $min|K_1\text{-}K_0|$ achieves the minimum average number of queries for n up to 4, it can be thought of as a lower bound on the average, and its gap between Teacher quantifies the benefits of knowing the actual function beforehand.

Figure 3 paints a clear picture of how the preexisting inference algorithms fare against each other. Hansel's algorithm was the best performer by far, Sokolov's came in second, and an algorithm using the subroutine FIND-BORDER (which is also used by Gainanov, 1984; Valiant, 1984; Makino and Ibaraki, 1995; Boros *et al.*, 1997) was a distant third. In fact, since the curve differences between Hansel and Sokolov, and Sokolov and the subroutine FIND-BORDER implementation, seem to increase with n, the corresponding difference in the average number of queries increases at rate greater than exponentially with n.

The difference between the curves for Hansel and "Teacher" decreases as n increases. The algorithm based on the criterion $min|K_1\text{-}K_0|$ has a curve that is almost parallel to Hansel's curve, indicating that the selection criterion

performs about 2% better than Hansel's algorithm. This decrease is especially clear in Figure 3 for n up to and including 8. For larger values of n, the high variance of our estimates makes it hard to distinguish the two curves, but the overall decreasing trends remain intact. It might seem that a 2% decrease is insignificant, but writing it as $2^n \times 0.02$ shows its real magnitude.

Another nice characteristic of this selection criterion is that it is the most consistent of all the algorithms. For example, it performs between 10 and 18 queries for 99.6% of the monotone Boolean functions in M_5. In contrast, the algorithm based on the subroutine FIND-BORDER is the least consistent with between 8 and 25 queries for 99.6% of the monotone Boolean functions.

4.2 Experimental Results for Problem 2

The results in Figures 4, 5, and 6 are based on an exhaustive analysis (i.e., all the monotone functions were generated) for n up to and including 4. For n = 4, 5, ...,12 random samples of functions were generated and the Horvitz-Thompson (1952) estimator is used to compute the averages for n greater than 4. The number of pairs of nested monotone Boolean functions generated were 2,000 for n = 5, 6, 7, and 200 for n = 8, 9, 10, and 100 for n = 11 and 12.

Figure 4 shows the average number of queries for Problem 2 when using the selection criteria. The lower curve corresponds to the unrestricted case (Problem 2.3), which achieves the fewest number of queries on the average. The sequential case (Problem 2.1), corresponding to the middle curve, is not as efficient as the unrestricted oracles in general, although they are very close for n = 1, 2, 3, and 4. The least efficient of the three types of oracles is the three-valued (Problem 2.2) corresponding to the upper curve.

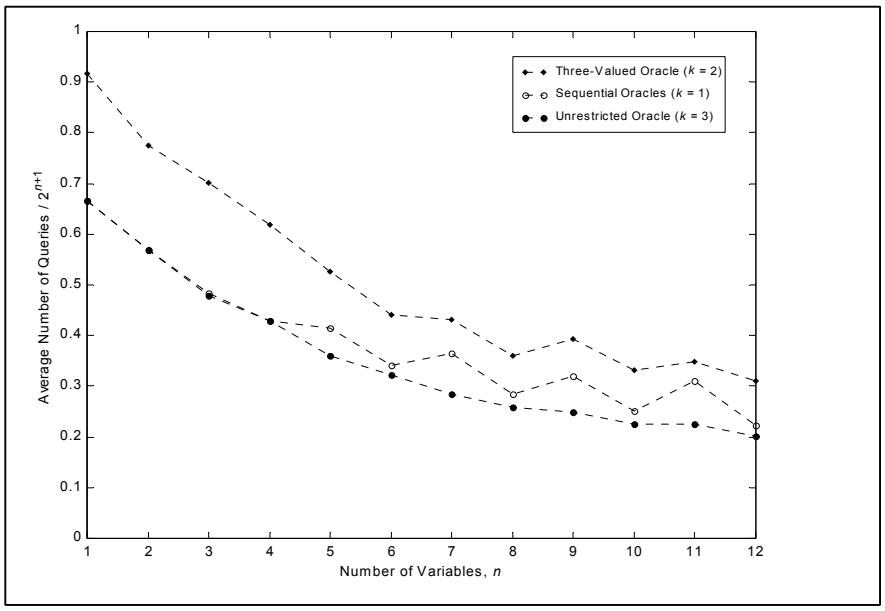
The gap between the curves in Figure 4 and the horizontal line *Average Number of Queries* / 2^{n+1} = 1 (the uppermost line of the box around the curves) can be thought of as the benefit of the monotone and nestedness assumptions together. This is due to the fact that 2^{n+1} is the number of required queries when the underlying pair of functions are neither nested or monotone. For example, when n = 12 in the unrestricted problem (k = 3) the average number of queries is reduced to about 20% of the maximum number of queries 2^{13} = 8,192 due to the monotone and nestedness assumptions.

Figure 5 quantifies the increase in the average number of queries due to the two restrictions on the oracles for n = 1,2, ..., 12. As mentioned earlier, the sequential oracles are practically unrestrictive for n = 1, 2, 3, and 4. For n greater than 4, the increase in average query complexity oscillates between 12% and 33% due to odd and even n, being much greater for odd n. In contrast, the three-valued oracle is much more restrictive across all the observed n, where the increase in the average number of queries oscillates between 35% and 55%, again due to odd and even n, being greater for odd n. In summary, the increases in the average number of queries for the sequential and three-valued cases are

dramatic. This is probably due to the fact that the average number of queries increases exponentially with the number of variables.

Figure 4. The Average Query Complexities for Problem 2.

If the nested property of the two functions defined on $\{0,1\}^n$ is ignored, the minimum total number of questions is, on average, $2Q(n)$. The benefit from the nestedness assumption for Problem 2 is quantified by the ratio of $Q_3(n)/2Q(n)$ which is given in Figure 6 for n = 1, 2, ..., 12. Therefore, the curves given in Figure 6 show the reduction in the average number of queries due to the nestedness assumption. This reduction decreases with the number of variables.



It starts out at 20% for n = 1, and oscillates between 1% and 10% for n greater than 7.

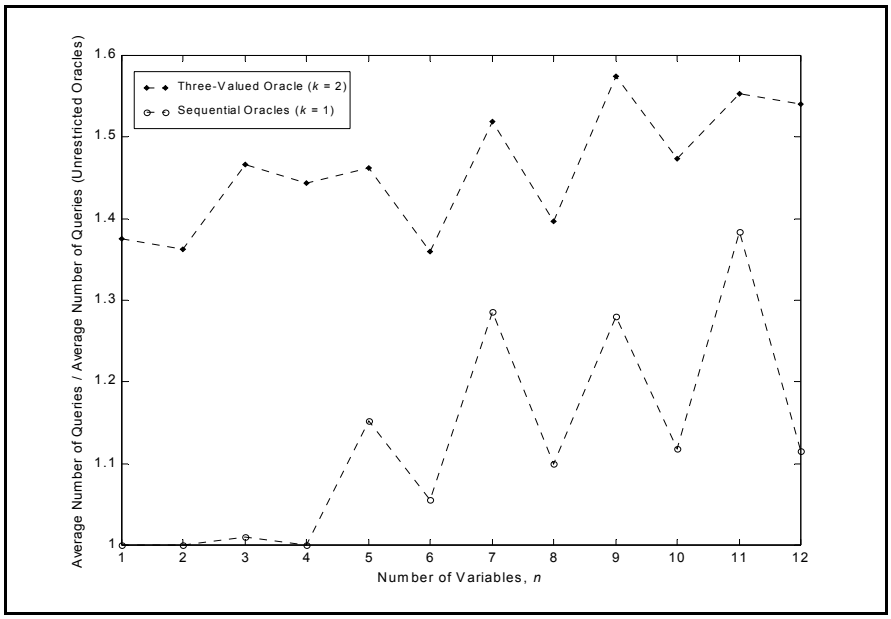
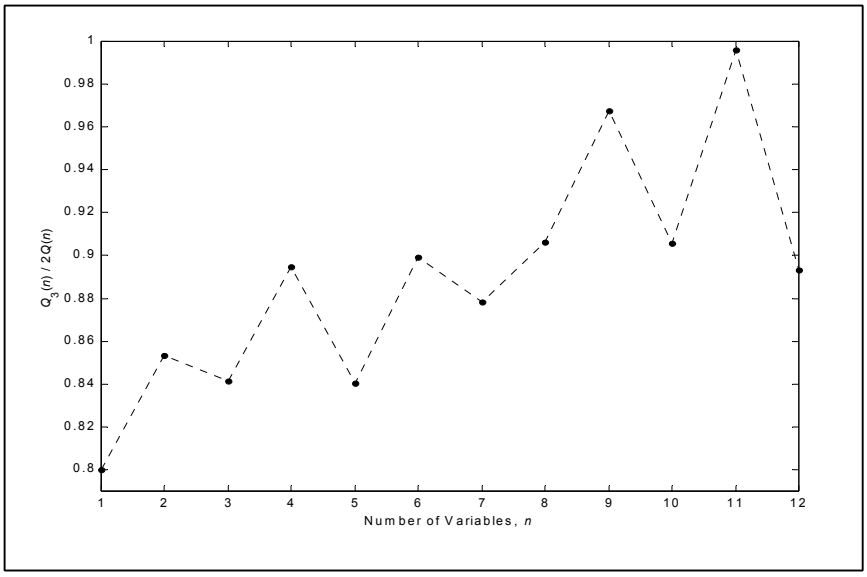


Figure 5. Increase in Query Complexities Due to Restricted Access to the Oracles.

Figure 6. Reduction in Query Complexity Due to the Nestedness Assumption.



4.3 Experimental Results for Problem 3

For the purpose of comparing the efficiency of the different selection criteria for Stage 3 on the same basis, ties resulting from the selection criteria ($min|K_0(v) - K_1(v)|$ for Stage 1, and $max(E_0(v) + E_1(v))$, max) 8(v), and v 0 LU(f*) c UZ(f*), the set of border vectors for Stage 3) were broken randomly. The four different inference processes using max) 8(v), v 0 LU(f*) c UZ(f*), $max(E_0(v) + E_1(v))$, or random selection for Stage 3 were simulated on the set of vertices $\{0,1\}^n$. For all three Stage 3 selection criteria, the selection criterion $min|K_0(v) - K_1(v)|$ was used for Stage 1 and random selection was used for Stage 2. The resulting simulations were repeated 100, 50, 25, and 10 times for each of 6 representative functions of M_n, with misclassification probabilities 0.1, 0.2, 0.3, and 0.4, for n = 2, 3, 4 and 5, respectively.

The representative functions are given in Table 5. For n = 4 and 5, these representative functions were randomly generated from a uniform distribution with individual probabilities of 1/Q(n) = 1/168 and 1/7581, respectively. For n = 3, the representative functions consist of non-similar functions (one from each similar subset of M_3). These functions represent all the functions in M_3, since the average case behavior is the same for a pair of similar monotone Boolean functions.

Table 5. The Representative Functions Used in the Simulations of Problem 3.

n = 2	n = 3	n = 4	n = 5
F	F	v_1v_2 W v_2v_4 W $v_1v_3v_4$	v_1v_4 W v_1v_5 W v_2v_4 W v_2v_5
v_1v_2	$v_1v_2v_3$	v_1v_2 W v_1v_3 W v_2v_3 W v_2v_4 W v_3v_4	v_1v_3 W v_2v_3 W v_2v_4 W $v_1v_2v_5$
v_1	v_1v_2	v_2v_3 W v_2v_4	v_2 W $v_1v_3v_4$ W $v_1v_4v_5$
v_2	v_1v_2 W v_1v_3	$v_1v_2v_3$ W $v_1v_3v_4$ W $v_2v_3v_4$	v_1v_3 W v_2v_4 W v_3v_5 W $v_1v_4v_5$
v_1 W v_2	v_1	v_1v_2 W v_2v_4 W v_3v_4	v_2v_4 W v_2v_5 W v_3v_5 W v_4v_5
T	v_1v_2 W v_1v_3 W v_2v_3	v_3 W v_1v_2 W v_1v_4	v_2v_5 W $v_1v_2v_3$ W $v_1v_3v_4$ W $v_1v_4v_5$ W $v_3v_4v_5$

To compute the overall average for a given q, the individual curves were weighted by the number of similar functions the representative function has (including itself) in M_3. The individual curves for the monotone Boolean functions F, $v_1v_2v_3$, v_1v_2, v_1v_2 W v_1v_3, v_1, and v_1v_2 W v_1v_3 W v_2v_3, were therefore weighted by 2, 2, 6, 6, 3, and 1, respectively. For n = 2, 4, and 5, the overall averages were computed without weights. The overall averages for n = 2 and 3 benefit from a reduced variance, since no additional errors are added due to the sampling of functions as done for n = 4 and 5.

Figure 7 shows the resulting average maximum likelihood curves for the inference problem defined on n = 2, 3, 4, and 5, and q = 0.1, 0.2, 0.3, and 0.4. Each curve is the average of 600, 300, 150, and 60 simulated inference processes observed for n = 2, 3, 4, and 5, respectively. In each plot, the horizontal axis corresponds to the number of Stage 3 queries, and the vertical axis corresponds to the maximum likelihood ratio. The curves are shown for the range of Stage 3 queries where the curves corresponding to the selection criterion *max*) 8(v) has a maximum likelihood ratio that is less than 0.99.

Not only do the curves corresponding to the guided selection criteria *max*) 8(v) and v 0 LU(f*) c UZ(f*) converge to 1 but they do so at a much faster rate than the curves corresponding to unguided random selection. In fact, the random selection achieves a maximum likelihood ratio of only about 0.7 after the same number of queries as the criterion *max*) 8(v) uses to reach 0.99, and the criterion v 0 LU(f*) c UZ(f*) uses to reach about 0.9, for n = 4.

The difference between the curves for unguided selection and these two guided selections grows with the misclassification probability q and with the dimension n. That is, the benefits from actively selecting vectors over passively receiving observations are greater when the values of q and n are large. In other words, the higher the misclassfication probability and the dimension of the problem are, the greater become the benefits of guiding the inference process.

The curves associated with criterion $max(E_0(v) + E_1(v))$ seems to converge to a value significantly less than 1. For example, when n = 3 and q = 0.3, the maximum likelihood ratio converges to about 0.4, and this value decreases as the values of q and n increase. Therefore, the larger error rate and the vector domain is, the more important it becomes to define an appropriate vector selection criterion.

Table 6 gives the average number of queries needed by the selection criterion *max*) 8(v) to converge to a maximum likelihood ratio of 0.99 for n =

2, 3, 4, and 5, and for q = 0.1, 0.2, 0.3, and 0.4. For a given n, these numbers increase dramatically as q increases. In fact, there seems to be more than a doubling in the numbers for fixed increments of q. For a given q, these numbers do not increase in such a dramatic fashion when n increases. However, they do increase faster than linearly with n.

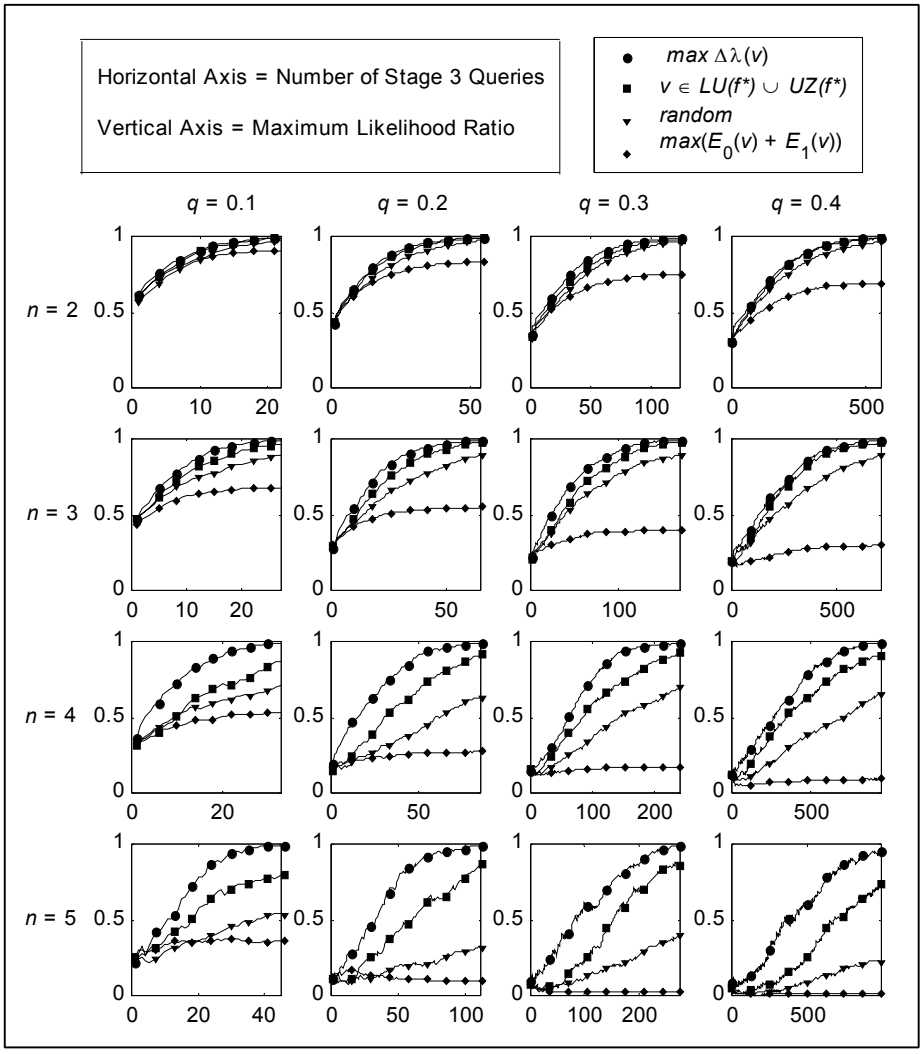


Table 6. The Average Number of Stage 3 Queries Used by the Selection Criterion *max*) 8(*v*) to Reach 8 > 0.99 in Problem 3 Defined on $\{0,1\}^n$ with Fixed Misclassification Probability *q*.

	q = 0.1	*q* = 0.2	*q* = 0.3	*q* = 0.4
n = 2	22	54	125	560
n = 3	27	65	170	710
n = 4	33	85	241	951
n = 5	45	111	277	1167

Randomly selecting the inferred border vectors (i.e., *v* 0 LU(*f**) c UZ(*f**)) makes the maximum likelihood ratio converge to 1, as long as the misclassification probabilities are all less than ½. That is, the misclassification probabilities do not necessarily have to be fixed. To see whether this holds for the selection criterion *max*) 8(*v*), consider an unrestricted model where the misclassification probability *q*(*v*) is a random variable distributed uniformly on the interval [*q*(1- *), *q*(1 + *)], where * 0 [0,1], for each vector *v* 0 $\{0,1\}^n$.

The case when * = 0 corresponds to the fixed misclassification probability model, that is, when *q*(*v*) is equal to *q* for all vectors *v* 0 $\{0,1\}^n$. The range of values for *q*(*v*) increases with *, but the expected value of *q*(*v*) is always equal to *q*. Therefore, the estimate of the maximum likelihood ratio based on the fixed *q* model is worse for larger values of *. To compare this estimate to an unrestricted estimate, the inference process was simulated 200 times for each * = 0, 0.5, and 1, holding constant *n* = 3 and the expected *q* = 0.2. Figure 8 shows the average maximum likelihood ratio curves for the unrestricted model (dotted curves) and the fixed model (solid curves) when using the selection criterion *max*) 8(*v*).

The regular and the unrestricted maximum likelihood ratios both converge to 1, though at slower rates as * increases. In other words, the selection criterion *max*) 8(*v*) is appropriate in situations where the misclassification probability is not necessarily fixed. In general, the unrestricted maximum likelihood ratio is much smaller than the regular one. For the case when *q*(*v*) is fixed at 0.2 (i.e., * = 0), the regular maximum likelihood ratio should be used, and when * > 0 it is an overestimate of the true maximum likelihood ratio. For the case when * = 1, the unrestricted maximum likelihood ratio should be used, and when * < 1 it may be an underestimate. The true likelihood ratio lies somewhere in between the two.

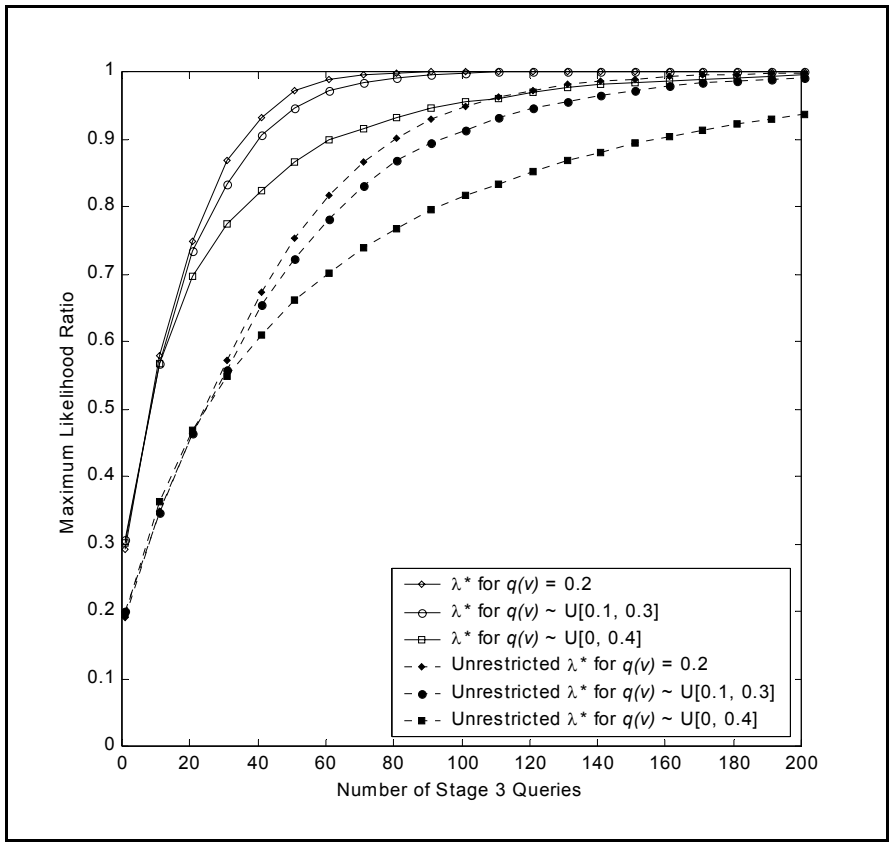


Figure 8. The Restricted and Regular Maximum Likelihood Ratios Simulated with Expected $q = 0.2$, and $n = 3$.

5. SUMMARY AND DISCUSSION

5.1 Summary of the Research Findings

The recent focus on the computational complexity has come at the expense of a drastic increase in the query complexity for Problem 1. In fact, the more recent the inference algorithm is, the worse it performs in terms of the average query complexity. The subroutine, here referred to as FIND-BORDER, is the most commonly used in the recent literature (Gainanov, 1984; Valiant, 1984; Makino and Ibaraki, 1995; Boros *et al.,*1997), and its performance was by far the worst. Therefore, the framework for unbiased empirical comparison of inference algorithms described in this chapter seems to be long overdue.

Even though guaranteeing the minimum average number of queries is currently only computationally feasible for relatively few variables (i.e., up to

5 or 6), the recursive algorithm used for Problem 1 revealed the non-intuitive nature of the optimal solutions. These solutions paved the way for the new selection criterion $min|K_1\text{-}K_0|$. This criterion would probably not have been developed (due to its non-intuitive nature) without the consultation of the optimal solutions.

The inference algorithm based on this selection criterion extends the feasible problem sizes to up to about 20 variables (which involves about 1 million vectors) for Problem 1. When the number of variables exceeds 20, computing the selection criterion might become intractable, while Hansel's algorithm will most likely still perform the best on the average. When creating the chain partition used in Hansel (1966) and Sokolov (1982) becomes intractable, perhaps finding border vectors one at a time by using the subroutine FIND-BORDER is still computationally feasible.

Problem 2 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 manyfold. First, it shows how the optimal and selection criterion approach to minimizing the average query complexity is extended to three different inference applications using a pair of nested monotone Boolean functions. The selection criteria seem to be good choices for the nested inference problem. They result in a slight increase in the average query complexity for the chain poset. For the poset $\{0,1\}^n$, 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 needed) and necessary (i.e., simpler models are not sufficient) for a wide variety of real world applications. Suppose a simpler model, such as a single monotone Boolean function, is used for these applications. At best, the simpler model will provide a poor approximation of the phenomenon under study. At worst, it will be unable to model the phenomenon. Suppose a more complex model, such as a pair of independent monotone Boolean functions, is used for these applications. Then, at the very least, the query complexity will increase. In addition, the inferred functions may lead to conflicting knowledge and are more likely to contain errors.

Third, it quantifies the reduction in query complexity due to the nestedness assumption. The improvement due to the nestedness assumption is between 6% and 8% for larger chain posets ($h > 50$). This improvement is greater for smaller chain posets, reaching its maximum of 20% for $h = 2$. In general, the average query complexity on the chain poset is $O(log(h))$, so this improvement is not very significant. For the poset $\{0,1\}^n$, this improvement is a few percent points for $n > 8$. This improvement decreases with the number of variables, reaching its maximum of 20% for $n = 1$. The average query complexity on the poset $\{0,1\}^n$ is exponential in n. This fact makes this improvement far more dramatic than for the chain poset.

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 84% and 55% increase in the average number of queries for the chain poset and the poset $\{0,1\}^n$, respectively. It is interesting to observe that the sequential oracles are just as efficient as the unrestricted oracles on the chain poset and for the poset $\{0,1\}^n$ for n up to 4. This implies that the pair of nested monotone Boolean functions defined on these posets can be inferred sequentially without losing optimality. For the poset $\{0,1\}^n$ with $n > 7$, the sequential oracle causes a significant increase in the average query complexity of 12-33%.

The maximum likelihood ratio approach to modeling the inference process of Problem 3 yielded a number of benefits. It was demonstrated that an appropriately defined guided learner, such as maximizing the expected maximum likelihood ratio (max) 8(v)) or randomly selecting inferred border vectors (v 0 LU(f*) c UZ(f*)), allowed the maximum likelihood ratio to converge to 1, even when the misclassification probability was not fixed. This avoids the bias problems associated with the variance approach reported in Cohn (1996), and also observed with the selection criterion $max(E_0(v) + E_1(v))$ which is based on the number of errors.

For complete reconstruction of monotone Boolean functions, the guided approach showed a dramatic reduction in the average number of queries over a passive learner. The simulations also indicated that this improvement grows at least exponentially as the number of variables n and the error rate q increase. Thus, defining an appropriate and efficient selection criterion is even more beneficial for large problems and applications with a high error rate.

For large problems (i.e., $n > 5$), it may not be possible to compute the selection criterion max) 8(v) since it takes exponential time (in the size of the query domain V) to do so. For such problems, queries can be selected randomly from the border vectors (v 0 LU(f*) c UZ(f*)). This only takes O(V) time, and results in much fewer queries than completely random selection on the average.

Hierarchical decomposition provides a way to address a large inference problem as a set of smaller independent inference problems. Even though it was not mentioned earlier, this decomposition is applicable to all three Problems 1, 2, and 3 where it can dramatically reduce the query complexity. Perhaps the greatest benefit of this decomposition is its simplified queries. This fact may not only improve the efficiency but also reduce the number of human errors, and hence increase the likelihood of inferring the correct function.

5.2 Significance of the Research Findings

The single most important discovery described in this chapter is the near optimal selection criteria which take polynomial time to evaluate. This leads to the efficient inference of monotone Boolean functions. The significance of these criteria is further strengthened by the scope of real-life problems that can be modeled by using monotone Boolean functions. Even though only one (or a pair of nested) monotone Boolean function(s) defined on the set of Boolean vectors $\{0,1\}^n$ were studied here, the selection criterion approach to guiding the learner is appropriate for any monotone mapping $V \to F$, where the sets $V \subseteq \{0,1\}^n$ and $F \subseteq \{0,1\}^r$ are both finite. The query domain can be viewed as a finite poset by using the monotonicity constraints: $f_i(v) \leq f_i(w)$ iff $v \preceq w$, for $I = 1, 2, ..., r$, and whatever the relationships between the functions are, such as the nestedness constraints: $f_1(v) \geq f_2(v) \forall v \in V$. The selection criteria can be evaluated for any such poset in order to pinpoint "smart" queries.

Once the border vectors have been established for each monotone function, they can be used to classify new observations. In addition, they can be represented by a (set of) monotone Boolean function(s) defined on a set of Boolean variables. Representing the inferred knowledge in this intuitive manner is perhaps the most important aspect of this problem when human interaction is involved since people tend to make better use of knowledge they can easily interpret, understand, validate, and remember.

The use of Boolean functions for analyzing fixed datasets has recently gained a momentum due their simple representation of intuitive knowledge. See Triantaphyllou and Soyster (1996b), Boros *et al.* (1995), Torvik *et al.* (1999), and Yilmaz *et al.* (2003) for example. Boolean models are also becoming more popular because methods for solving their related hard logical optimization problems are emerging (e.g., Triantaphyllou (1994), Chandru and Hooker (1999), Hooker (2000), and Felici and Truemper (2002)). Some initial studies on guided inference of Boolean functions from fixed datasets are provided in Triantaphyllou and Soyster (1996a) and Nieto-Sanchez *et al.* (2002).

The narrow vicinity hypothesis proposed by Kovalerchuk *et al.* (2000a) suggests that the use of the monotonicity assumption is often necessary and sufficient. As such, it can greatly improve upon knowledge representations that are too simple or too complex. This chapter demonstrated that the problem of guided inference in the presence of monotonicity can be of great benefit in a wide variety of important real-life applications.

5.3 Future Research Directions

As mentioned in section 5.2 the selection criterion approach to learning monotone Boolean functions defined on $\{0,1\}^n$ is applicable in the much more

general monotone setting: $V \rightarrow F$, where the sets $V \subseteq \mathbb{R}^n$ and $F \subseteq \mathbb{R}^r$ are both finite. The monotone mapping $V \rightarrow F$, where the set $V \subseteq \mathbb{R}^n$ is infinite and the set $F \subseteq \mathbb{R}^r$ 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 too. Let U denote the unclassified set (i.e., a subset of V) and let the parameters $K_0(v)$ and $K_1(v)$ now denote the size of the subsets $\{w \in U: w \succeq v\}$ and $\{w \in U: v \succeq w\}$, respectively. For example, $K_z(v)$ is measured in terms of distance, 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.

For the problems considered in this chapter, the selection criteria attempt to minimize the average query costs. This objective is based on certain assumptions of the query costs (fixed cost of querying an oracle in Problems 1, 2, and 3, and highly disproportionate or equal query costs for the two oracles in Problems 2.1 and 2.3, respectively). It would be interesting to see how the dialogue with the oracle(s) changes as these assumptions are modified. When dealing with two oracles, it may be that the cost of querying the first oracle may be less than, yet of similar magnitude as, the cost of querying the second oracle. In this case, the first few queries should be directed at the first oracle. After a few queries it may be cost beneficial to begin alternating between the two oracles. It could also be that the order of the queries has an effect on the total inference cost. In some applications, additional properties may be known about the underlying function. Some applications may put a limit on the number of lower units, shifting the focus of the optimal vertices from the vertical center to the vertical edge of the poset. It may be that the underlying function belongs to a subclass of monotone Boolean functions, such as threshold functions, 2-monotonic functions, etc.

6. CONCLUDING REMARKS

The methodologies presented in this chapter provide a framework for solving diverse and potentially very important real-life problems that can be modeled as guided inference problems in the presence of monotonicity. The benefits of these methodologies were shown to be dramatic for the specific problems studied here. However, these research findings are just the tip of the iceberg. The interested reader is referred to Torvik and Triantaphyllou (2002, 2003, 2004) for further details on the methodology for Problems 1, 2, and 3, respectively.

ACKNOWLEDGMENTS

The authors are very appreciative for the support by the U.S. Navy, Office of Naval Research (ONR), research grants N00014-95-1-0639 and N00014-97-1-0632.

REFERENCES

M. Ayer, H.D. Brunk, G.M. Ewing, W.T. Reid, and E. Silverman, "An Empirical Distribution Function for Sampling with Incomplete Information," *Annals of Mathematical Statistics*, Vol. 26, pp. 641-647, 1955.

A. Ben-David. "Automatic Generation of Symbolic Multiattribute Ordinal Knowledge-Based DSSs: Methodology and Applications," *Decision Sciences*, Vol. 23, No. 6, pp. 1357-1372, 1992.

A. Ben-David, "Monotonicity Maintenance in Information-Theoretic Machine Learning Algorithms," *Machine Learning*, Vol. 19, No. 1, pp. 29-43, 1995.

J.C. Bioch and T. Ibaraki, "Complexity of Identification and Dualization of Positive Boolean Functions," *Information and Computation*, Vol. 123 pp. 50-63, 1995.

D.A. Bloch and B. W. Silverman, "Monotone Discriminant Functions and Their Applications in Rheumatology," *Journal of the American Statistical Association*, Vol. 92, No. 437, pp. 144-153, 1997.

H. Block, S. Qian, and A. Sampson, "Structure Algorithms for Partially Ordered Isotonic Regression," *Journal of Computational and Graphical Statistics*, Vol. 3, No. 3, pp. 285-300, 1994.

E. Boros, P.L. Hammer, and J.N. Hooker, "Predicting Cause-Effect Relationships from Incomplete Discrete Observations," *SIAM Journal on Discrete Mathematics*, Vol. 7, No. 4, pp. 531-543, 1994.

E. Boros, P.L. Hammer, and J.N. Hooker, "Boolean Regression," *Annals of Operations Research*, Vol. 58, pp. 201-226, 1995.

E. Boros, P.L. Hammer, T. Ibaraki., and K. Makino, "Polynomial-Time Recognition of 2-Monotonic Positive Boolean Functions Given by an Oracle," *SIAM Journal on Computing*, Vol. 26, No. 1, pp. 93-109, 1997.

V. Chandru and J.N. Hooker, "Optimization Methods for Logical Inference," John Wiley & Sons, New York, NY, USA, 1999.

R. Church, " Numerical Analysis of Certain Free Distributive Structures," *Duke Mathematical Journal*, Vol. 6, pp. 732-734, 1940.

R. Church, "Enumeration by Rank of the Free Distributive Lattice with 7 Generators," *Notices of the American Mathematical Society*, Vol. 11 pp. 724, 1965.

D.A. Cohn, "Neural Network Exploration Using Optimal Experiment Design," *Neural Networks*, Vol. 9, No. 6, pp. 1071-1083, 1996.

D.A. Cohn, "Minimizing Statistical Bias with Queries," A.I. Memo No. 1552, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA, 1995.

T.H. Cormen, C.H. Leiserson, and R.L. Rivest, "*Introduction to Algorithms*," The MIT Press, Cambridge, MA, USA, 1997.

R. Dedekind, R, "Ueber Zerlegungen von Zahlen durch ihre Grössten Gemeinsamen Teiler," *Festschrift Hoch. Brauhnschweig u. ges Werke II*, pp. 103-148, 1897.

T. Eiter and G. Gottlob, "Identifying the Minimal Transversals of a Hypergraph and Related Problems," *SIAM Journal on Computing*, Vol. 24, No. 6, pp. 1278-1304, 1995.

K. Engel. *Encyclopedia of Mathematics and its Applications 65: Sperner Theory*," Cambridge University Press, Cambridge, MA, USA, 1997.

V.V. Federov, "*Theory of Optimal Experiments*," Academic Press, New York, NY, USA, 1972.

G. Felici and K. Truemper, "A MINSAT Approach for Learning in Logic Domains," *INFORMS Journal on Computing*, Vol. 14, No. 1, pp. 20-36, 2002.

M.L. Fredman and L. Khachiyan, "On the Complexity of Dualization of Monotone Disjunctive Normal Forms," *Journal of Algorithms*, Vol. 21, pp. 618-628, 1996.

D.N. Gainanov, "On One Criterion of the Optimality of an Algorithm for Evaluating Monotonic Boolean Functions,"*U.S.S.R. Computational Mathematics and Mathematical Physics*, Vol. 24, No. 4, pp. 176-181, 1984.

G. Hansel, "Sur Le Nombre Des Foncions Booleenes Monotones De n Variables," *C. R. Acad. Sc. Paris*, Vol. 262, pp. 1088-1090, 1966.

J.N. Hooker, "Logic Based Methods for Optimization," John Wiley & Sons, New York, NY, USA, 2000.

D.G. Horvitz and D.J. Thompson, "A Generalization of Sampling without Replacement from a Finite Universe," *Journal of the American Statistical Association*, Vol. 47, pp. 663-685, 1952.

D.H. Judson, "On the Inference of Semi-coherent Structures from Data," A Master's Thesis, University of Nevada, Reno, NV, USA, 1999.

D.H. Judson, "A Partial Order Approach to Record Linkage," Federal Committee on Statistical Methodology Conference, November 14-16, Arlington, VA, USA, 2001.

A.V. Karzanov, "Determining the Maximal Flow in a Network by the Method of Preflows," *Soviet Mathematics Doklady*, Vol. 15, pp. 434-437, 1974.

A.D. Korshunov, On the Number of Monotone Boolean Functions," *Problemy Kibernetiki*, Vol. 38, pp. 5-108, 1981 (in Russian).

B. Kovalerchuk, E. Triantaphyllou, and A.S. Deshpande, "Interactive Learning of Monotone Boolean Functions," *Information Sciences*, Vol. 94, pp. 87-118, 1996.

B. Kovalerchuk, E. Triantaphyllou, J.F. Ruiz, V.I. Torvik, and E. Vitayev, "The Reliability Issue of Computer-Aided Breast Cancer Diagnosis," *Computers and Biomedical Research*, Vol. 33, pp. 296-313, 2000a.

B. Kovalerchuk, B. and E. Vityaev, "*Data Mining in Finance*," Kluwer Academic Publishers, Boston, MA, USA, 2000b.

C.I.C. Lee, "The min-max Algorithm and Isotonic Regression," *The Annals of Statistics*, Vol. 11, pp. 467-477, 1983.

D.J.C. MacKay, "Information-based Objective Functions for Active Data Selection," *Neural Computation*, Vol. 4, No. 4, pp. 589-603, 1992.

K. Makino and T. Ibaraki, "A Fast and Simple Algorithm for Identifying 2-Monotonic Positive

Boolean Functions," *Proceedings of ISAACS'95, Algorithms and Computation*, Springer-Verlag, Berlin, Germany, pp. 291-300, 1995.

K. Makino and T. Ibaraki, "The Maximum Latency and Identification of Positive Boolean Functions," *SIAM Journal on Computing*, Vol. 26, No. 5 , pp. 1363-1383, 1997.

K. Makino, T. Suda, H. Ono, and T. Ibaraki, "Data Analysis by Positive Decision Trees. *IEICE Transactions on Information and Systems* , Vol. E82-D, No. 1, pp. 76-88, 1999.

S. Nieto-Sanchez, E. Triantaphyllou, J. Chen, and T.W. Liao, "An Incremental Learning Algorithm for Constructing Boolean Functions From Positive and Negative Examples," *Computers and Operations Research*, Vol. 29, No. 12, pp. 1677-1700, 2002.

J.C. Picard, "Maximal Closure of a Graph and Applications to Combinatorial Problems," *Management Science*, Vol. 22, pp. 1268-1272, 1976.

T. Robertson, F.T. Wright, and R.L. Dykstra, "*Order Restricted Statistical Inference*. John Wiley & Sons, New York, NY, USA, 1988

I. Shmulevich, "Properties and Applications of Monotone Boolean Functions and Stack Filters," A Ph.D. Dissertation, Department of Electrical Engineering, Purdue University, West Lafayette, IN, USA, 1997.

N.A. Sokolov, "On the Optimal Evaluation of Monotonic Boolean Functions," *U.S.S.R. Computational Mathematics and Mathematical Physics*," Vol. 22, No. 2, pp. 207-220, 1982.

C. Tatsuoka and T. Ferguson, "Sequential Classification on Partially Ordered Sets," Technical Report 99-05, Department of Statistics, The George Washington University, Washington, D.C., USA, 1999.

E. Triantaphyllou, "Inference of a Minimum Size Boolean Function by Using a New Efficient Branch-and-Bound Approach from Examples," *Journal of Global Optimization*, Vol. 5 , pp. 69-84, 1994.

E. Triantaphyllou and A.L. Soyster, "An Approach to Guided Learning of Boolean Functions," *Mathematical and Computer Modelling*, Vol. 23, No. 3, pp 69-86, 1996a.

E. Triantaphyllou and A.L. Soyster, "On the Minimum Number of Logical Clauses Which Can be Inferred From Examples," *Computers and Operations Research*, Vol. 23, No. 8, pp. 783-799, 1996b.

V.I. Torvik, E. Triantaphyllou, T.W. Liao and S.W. Waly, "Predicting Muscle Fatigue via Electromyography: A Comparative Study," *Proceedings of the 25th International Conference of Computers and Industrial Engineering*, pp. 277-280, 1999.

V.I. Torvik and E. Triantaphyllou, "Minimizing the Average Query Complexity of Learning Monotone Boolean Functions," *INFORMS Journal on Computing*, Vol. 14, No. 2, pp. 144-174, 2002.

V.I. Torvik and E. Triantaphyllou, "Guided Inference of Nested Monotone Boolean Functions. *Information Sciences*, Vol. 151, 171-200, 2003.

V.I. Torvik, M. Weeber, D.R. Swanson, and N.R. Smalheiser, "A Probabilistic Similarity Metric for Medline Records: A Model for Author Name Disambiguation," To appear in *JASIST*, 2004.

V.I. Torvik and E. Triantaphyllou, "Guided Inference of Stochastic Monotone Boolean Functions", Under review 2004.

L.G. Valiant, "A Theory of the Learnable," *Communications of the ACM*, Vol. 27, No. 11, pp.

1134-1142, 1984.

M. Ward, "Note on the Order of the Free Distributive Lattice," *Bulletin of the American Mathematical Society*, Vol. 52, No. 135, pp. 423, 1946.

D. Wiedemann, "A Computation of the Eight Dedekind Number," *Order*, Vol. 8, pp. 5-6, 1991.

E. Yilmaz, E. Triantaphyllou, J. Chen, and T.W. Liao, "A Heuristic for Mining Association Rules In Polynomial Time," *Mathematical and Computer Modelling*, Vol. 37, No. 1-2, pp. 219-233, 2003.

AUTHORS' BIOGRAPHICAL STATEMENTS

Dr. Torvik is a Research Assistant Professor in the Department of Psychiatry at the University of Illinois at Chicago, where he is managing a literature-based knowledge discovery project called Arrowsmith (jointly funded by the National Library of Medicine and the National Institute of Mental Health; PI: Neil R. Smalheiser, M.D., Ph.D.). He received his B.A. in Mathematics from St. Olaf College in Northfield, MN in 1995, his M.S. in Operations Research from the Oregon State University in Corvallis, OR in 1997, and his Ph.D. in Engineering Science from the Louisiana State University in Baton Rouge, LA in 2002. His Ph.D. dissertation titled "Data Mining and Knowledge Discovery: A Guided Approach Based on Monotone Boolean Functions" was awarded the 2002 LSU Distinguished Dissertation Award. His research interests include mathematical optimization and computational statistics applied to literature-based knowledge discovery and bioinformatics.

Dr. Triantaphyllou did his graduate studies at Penn State University from 1984 to 1990. While at Penn State, he earned a Dual M.S. degree in Environment and Operations Research (OR), an M.S. degree in Computer Science and a Dual Ph.D. degree in Industrial Engineering and Operations Research. Since the spring of 2005 he is a Professor in the Computer Science Department at the Louisiana State University (LSU) in Baton Rouge, LA, U.S.A., after he has served for 11 years as an Assistant, Associate, and Full Professor in the Industrial Engineering Department at the same university. He has also served for one year as an Interim Associate Dean for the College of Engineering at LSU.

His research is focused on decision-making theory and applications, data mining and knowledge discovery, and the interface of operations research and computer science. Since the years he was a graduate student, he has developed new methods for data mining and knowledge discovery and also has explored some of the most fundamental and intriguing subjects in decision making. In 1999 he has received the prestigious IIE (Institute of Industrial Engineers), OR Division, Research Award for his research contributions in the above fields. Some of his graduate students have also received awards and distinctions including the Best Dissertation Award at LSU for Science, Engineering and Technology for the year 2003. In 2000 Dr. Triantaphyllou published a bestseller book on multi-criteria decision-making. Also, in 2005 he published a monograph on data mining and knowledge discovery, besides co-editing a book on the same subject.

He always enjoys sharing the results of his research with his students and is also getting them actively involved in his research activities. He has received teaching awards and distinctions. His research has been funded by federal and

state agencies, and the private sector. He has extensively published in some of the top refereed journals and made numerous presentations in national and international conferences.

Dr. Triantaphyllou has a strong inter-disciplinary background. He has always enjoyed organizing multi-disciplinary teams of researchers and practitioners with complementary expertise. These groups try to comprehensively attack some of the most urgent problems in the sciences and engineering. He is a strong believer of the premise that the next round of major scientific and engineering discoveries will come from the work of such inter-disciplinary groups. More details of his work can be found in his web site *(http://www.imse.lsu.edu/vangelis/*).