NORTH-HOLLAND

Intelligent Systems

Interactive Learning of Monotone Boolean Functions

BORIS KOVALERCHUK
EVANGELOS TRIANTAPHYLLOU
ANIRUDDHA S. DESHPANDE

Department of Industrial and Manufacturing Systems Engineering, 3128 CEBA Building, Louisiana State University, Baton Rouge, Louisiana 70803-6409

and

EVGENII VITYAEV

Institute of Mathematics, Russian Academy of Science, Novosibirsk, 630090, Russia

Communicated by Subhash Kak and George M. Georgiou

ABSTRACT

This paper presents some optimal interactive algorithms for some problems related to learning of monotone Boolean functions. These algorithms are based on the fundamental Hansel theorem. The advantage of the algorithms is that they are not heuristics, as is often the case of many known algorithms for general Boolean functions, but they are optimal in the sense of the Shannon function. This paper also formulates a new problem for the joint restoration of two nested monotone Boolean functions f_1 and f_2. This formulation allows one to further decrease the dialogue with an expert and restore nonmonotone functions of the form $f_2 \& \rceil f_1$. The effectiveness of the proposed approaches is demonstrated by some illustrative computational experiments.

1. INTRODUCTION

An important problem in machine learning is the one which involves the inference of a Boolean function from collections of positive and negative examples. This is also called the *logical analysis problem* [13] and is a special case of inductive inference. This kind of knowledge extraction is desirable when one is interested in deriving a set of rules which, in turn, can be easily comprehended by a field expert. In many application do-

INFORMATION SCIENCES **94**, 87–118 (1996)

655 Avenue of the Americas, New York, NY 10010

0020-0255/96/$15.00
PII S0020-0255(96)00082-5

mains, the end users do not have sophisticated computer and modelling expertise. As a result, systems which are based on techniques such as neural networks, statistics, or linear programming are not appealing to them, not do these methods provide a plausible explanation of the underlying decision-making process. On the other hand, a logical analysis approach, when it is applicable, can result in rules which are already known to the end user (thus increasing his/her confidence in the method) or lead to new discoveries. In other words, a logical analysis approach has the distinct advantage of high comprehensibility when it is compared with other methods and to the discovery of new explicit knowledge. This is an important step on the way to scientific explanation of a studied phenomenon, in comparison with a "black box" type of approach.

The most recent advances in distinguishing between elements of two pattern sets can be classified into six distinct categories. These are a clause satisfiability approach to inductive inference by Kamath et al. [18, 19]; some branch-and-bound approaches of generating a small set of logical rules by Triantaphyllou et al. [34] and Triantaphyllou [35]; some improved polynomial time and NP-complete cases of Boolean function decomposition by Boros et al. [6]; linear programming approaches by Woldberg and Mangasarian [40], and Mangasarian et al. [27]; some knowledge-based learning approaches combining symbolic and connectionist (neural networks) machine-based learning as proposed by Shavlik [33], Fu [10], Goldman et al. (1994), and Cohn et al. [7]; and finally, some nearest-neighbor classification approaches by Hattoru and Torii [17], Kurita [26], and Kamgar-Parsi and Kanal [20]. From the above six categories, the first three can be considered as logical analysis approaches, since they deal with inference of Boolean functions.

The general approach of pure machine learning and inductive inference includes the following two steps: (i) obtaining in advance a sufficient number of examples (vectors) for different classes of observations, and (ii) formulation of the assumptions about the required mathematical structure of the example population (see, for instance, [5, 42, 9, 39]). Human interaction is used just when one obtains new examples and formulates the assumptions.

The general problem of learning a Boolean function has many applications. Such applications can be found in the areas of medical diagnosis, hardware diagnosis, astrophysics, and finance, among others, as is best demonstrated by the plethora of databases in the Machine Learning Repository in the University of California at Irvine [29].

However, traditional machine learning approaches have some difficulties. In particular, the size of the hypothesis space is influential in determining the *sample complexity* of a learning algorithm, that is, the expected

number of examples needed to accurately approximate a target concept. The presence of *bias* in the selection of a hypothesis from the hypothesis space can be beneficial in reducing the sample complexity of a learning algorithm [28, 30, 11]. Usually the amount of bias in the hypothesis space H is measured in terms of the *Vapnik-Chervonenkis dimension*, denoted as $VCdim(H)$ [37, 15]. Theoretical results regarding the $VCdim(H)$ are well known [37]. The results in [37] are still better than some other bounds given in [4]. However, all these bounds are still overestimates [16].

The learning problem examined in this paper is how one can infer a Boolean function. We assume that initially some Boolean vectors (input examples) are available. These vectors are defined in the space $\{0,1\}^n$, where n is the number of binary attributes or atoms. Each such vector represents either a positive or a negative example, depending on whether it must be accepted or rejected by the target Boolean function. In an interactive environment, we assume that the user starts with an initial set (which may be empty) of positive and negative examples, and then he/she asks an *oracle* for membership classification of new examples which are selected according to some interactive guided learning strategy [32, 1]. In [36], there is a discussion of this issue, and a guided learning strategy for general Boolean functions is presented and analyzed.

The main challenge in inferring a target Boolean function from positive and negative examples is that the user can never be absolutely certain about the correctness of the inferred function, unless he/she has used the entire set of all possible examples, which is of size 2^n. Apparently, even for a small value of n, this task may be practically impossible to realize.

Fortunately, many real-life applications are governed by the behavior of a *monotone* system or can be described by a *combination of a small number of monotone systems*. Roughly speaking, monotonicity for the one-dimensional case means that the value of the output increases or decreases when the value of the input increases. A formal definition of this concept for the multidimensional case is provided in the next section. This is common, for example, in many medical applications, where, for instance, the severity of a condition directly depends on the magnitude of the blood pressure and body temperature within certain intervals.

In machine learning, monotonicity offers some unique computational advantages [38, 2]. By knowing the class of certain examples, one can easily infer the class membership of more examples. This, in turn, can significantly expedite the learning process.

This paper is organized as follows. Section 2 provides some basic definitions and results about monotone Boolean functions. Section 3 presents some key research problems with highlights of some possible solution approaches. Section 4 describes some procedures for inferring

monotone Boolean functions. Sections 5 and 6 illustrate the previous procedures in terms of an application. Finally, the paper ends with some concluding remarks and suggestions for possible extensions.

2. SOME BASIC DEFINITIONS AND RESULTS ABOUT MONOTONE BOOLEAN FUNCTIONS

Let E_n denote the set of all binary vectors of length n. Let x and y be two such vectors. Then, the vector $x=(x_1, x_2, x_3, \ldots, x_n)$ *precedes* the vector $y=(y_1, y_2, y_3, \ldots, y_n)$ (denoted as $x \succcurlyeq y$) if and only if the following is true: $x_i \leq y_i$, for all $1 \leq i \leq n$. If, at the same time, $x \neq y$, then it is said that x *strictly* precedes y (denoted as $x \prec y$). The two binary vectors x and y are said to be *comparable* if one of the relationships $x \preccurlyeq y$ or $x \succcurlyeq y$ holds.

A Boolean function $f(x)$ is *monotone* if, for any vectors $x, y \in E_n$, the relation $f(x) \leq f(y)$ follows from the fact that $x \preccurlyeq y$. Let M_n be the set of all monotone Boolean functions defined on n variables. A binary vector x of length n is said to be an *upper zero* of a function $f(x) \in M_n$, if $f(x)=0$ and for any vector y such that $x \prec y$, we have $f(y)=1$. Also, we shall call the number of unities (i.e., the number of the "1" elements) in vector x as its *level* and denote this by $U(x)$. An upper zero x of a function f is said to be the *maximal upper zero* if we have $U(y) \leq U(x)$ for any upper zero y of the function f (e.g., [23]). In an analogous manner, we can define the concepts of lower unit and minimal lower unit. A binary vector x of length n is said to be a *lower unit* of a function $f(x) \in M_n$, if $f(x)=1$ and, for any vector y from E_n such that $y \prec x$, we get $f(y)=0$. A lower unit of a function f is said to be the *minimal lower unit* if we have $U(x) \leq U(y)$ for any lower unit y of the function f.

Examples of monotone Boolean functions are the constants 0 and 1, the identity function $f(x)=x$, the disjunction $(x_1 \vee x_2)$, the conjunction $(x_1 \wedge x_2)$, etc. Any function obtained by a composition of monotone Boolean functions is itself monotone. In other words, the *class* of all monotone Boolean functions is *closed*. Moreover, the class of all monotone Boolean functions is one of the five *maximal* (*precomplete*) *classes* in the set of all Boolean functions. That is, there is no closed class of Boolean functions, containing all monotone Boolean functions and distinct from the class of monotone functions and the class of all Boolean functions. The reduced *disjunctive normal form* (DNF) of any monotone Boolean function, distinct of 0 and 1, does not contain negations of variables. The set of functions $\{0, 1, (x_1 \vee x_2), (x_1 \wedge x_2)\}$ is a *complete system* (and moreover, a *basis*) in the class of all monotone Boolean functions [3].

For the *number* $\psi(n)$ of *monotone Boolean functions* depending on n variables, it is known that

$$\Psi(n) = 2^{\binom{n}{\lfloor n/2 \rfloor}(1+\epsilon(n))},$$

where $0 < \epsilon(n) < c(\log n)/n$ and c is a constant (see, for instance, [21, 3]). Let a monotone Boolean function f be defined with the help of a certain operator A_f (also called an *oracle*) which, when fed with a vector $x = (x_1, x_2, x_3, \ldots, x_n)$, returns the value of $f(x)$. Let $\mathcal{F} = \{F\}$ be the set of all algorithms which can solve the above problem, and $\varphi(F, f)$ be the number of accesses to the operator A_f required to solve a given problem about inferring a monotone function $f \in M_n$.

Next, we introduce the Shannon function $\varphi(n)$ as follows [22]:

$$\varphi(n) = \min_{F \in \mathcal{F}} \max_{f \in M_n} \varphi(F, f). \tag{2.1}$$

The problem examined next is that of finding all maximal upper zeros (lower units) of an arbitrary function $f \in M_n$ with the help of a certain number of accesses to the operator A_f. It is shown in [14] that in the case of this problem, the following relation is true (known as *Hansel's theorem*):

$$\varphi(n) = \binom{n}{\lfloor n/2 \rfloor} + \binom{n}{\lfloor n/2 \rfloor + 1}. \tag{2.2}$$

Here $\lfloor n/2 \rfloor$ is the closest integer number to $n/2$ which is no greater than $n/2$. In terms of machine learning, the set of all maximal upper zeros represents the *border elements* of the negative pattern. In an analogous manner, the set of all minimal lower units represents the border of a positive pattern. In this way, a monotone Boolean function represents two *compact patterns*.

Restoration algorithms for monotone Boolean functions which use Hansel's theorem are optimal in terms of the Shannon function. That is, they minimize the maximum time requirements of any possible restoration algorithm. It is interesting to note at this point that, to the best of our knowledge, Hansel's theorem has not been translated into English, although there are numerous references of it in the non-English literature

(Hansel wrote his paper in French). This theorem is one of the final results of the long-term efforts in monotone Boolean functions beginning from Dedekind [8] in 1897.

One approach to using monotonicity in combination with existing pattern recognition/classification approaches is to consider each one of the available examples and apply the concept of monotonicity to generate many more data. However, such an approach would have to *explicitly* consider a potentially huge set of input observations. On the other hand, these classification approaches are mostly NP-complete and very CPU time-consuming. Thus, such an approach would be inefficient.

In [12], an approach for restoring a single monotone Boolean function is presented in connection with a pattern recognition/classification problem. That approach implicitly considers all derivable data from a single observation by utilizing Hansel chains of data [14]. That algorithm is optimal in the sense of the Shannon function. However, although this result has also been known in the non-English literature (it was originally published in Russian), to the best of our knowledge it has not yet been described in the English literature.

We will call a montone Boolean function an *increasing* (*isotone*) *monotone* Boolean function in contrast with a decreasing monotone Boolean function. A Boolean function is *decreasing* (*antitone*) *monotone* if, for any vectors $x, y \in E_n$, the relation $f(x) \leq f(y)$ follows from the fact that $x \geq y$ [31, p. 149].

Each general discrimination (i.e., a general Boolean function) can be described in terms of several increasing $g_i(x_1, \ldots, x_n)$ and decreasing $h_i(x_1, \ldots, x_n)$ monotone Boolean functions [24]. That is, the following is always true:

$$q(x) = \bigvee_{j=1}^{m} \left(g_j(x) \wedge h_j(x) \right). \tag{2.3}$$

Next, let us consider the case in which $q(x) = g(x) \wedge h(x)$. Here, $q^+ = g^+ \cap h^+$, where $q^+ = \{x: q(x) = 1\}$, $g^+ = \{x: g(x) = 1\}$, and $h^+ = \{x: h(x) = 1\}$. Therefore, one can obtain the set of all positive examples for q as the intersection of the sets of all positive examples for the monotone functions g and h.

For a general function $q(x)$, represented as in (2.3), the union of all these intersections gives the full set of positive examples: $q^+ = \cup q_j^+ = \cup (g_j^+ \cap h_j^+)$. Often, we do not need so many separate monotone functions. The union of all conjunctions which do not include negations $\bar{x}_i$ forms a single increasing monotone Boolean function (see, for instance, [41, 3]).

3. SOME KEY PROBLEMS AND ALGORITHMS

In this section we present some key problems and the main steps of algorithms for solving them.

PROBLEM 1. Inference of a monotone function with no initial data.

Conditions. There are no initial examples to initiate learning. All examples should be obtained as a result of the interaction of the designer with an "oracle" (i.e., an operator A_f). It is also required that the discriminant function should be a monotone Boolean function. This problem is equivalent to the requirement that we consider only two compact monotone patterns. These conditions are natural in many applications. Such applications include the estimation of reliability (see also the illustrative examples next in this section for Problem 2).

ALGORITHM A1:

Step 1: The user is asked to confirm function monotonicity.

Step 2: Apply an iterative algorithm for generating examples and construction of the DNF representation. Do so by using Hansel's lemma (to be described in Section 4; see also [14, 12]). This algorithm is optimal according to relations (2.1) and (2.2).

PROBLEM 2. The nested classification problem.

Conditions. When we have one classification problem, we should learn to classify an arbitrary vector as a result of the interaction of the designer with an "oracle" (i.e., an operator A_f). Now we assume that we have two classification problems, i.e., we need to restore two Boolean functions f_1 and f_2, to discriminate classes. We want to solve these classification problems simultaneously, by interacting with two oracles A_{f1} and A_{f2}. We also assume that these two problems are *nested*, i.e., all positive cases for f_1 are also positive cases for f_2. That is, for all $\alpha \in E_n$, the following relation is always true: $f_1(\alpha) \geq f_2(\alpha)$. We also accompany this assumption with the requirement of monotonicity of f_1 and f_2.

This situation is more complex than the one for Problem 1. However, the use of additional information from both problems allows for the potential to accelerate the rate of learning.

Some Examples of Nested Problems

First illustrative example: The engineering reliability problem For illustrative purposes, consider the problem of classifying the states of some system by a reliability related expert. This expert is assumed to have worked with

this particular system for a long term and thus can serve as an "oracle" (i.e., the operator A_f). States of the system are represented by binary vectors from E_n (binary space defined on n 0-1 attributes or characteristics). The "oracle" is assumed that can answer questions such as: "*Is reliability of a given state guaranteed*?" (Yes/No) or: "*Is an accident for a given state guaranteed*?" (Yes/No). In accordance with these questions, we pose *two* interrelated nested classification tasks. The first one is for answering the first question, while the second task is for answering the second question. Next, we define the four possible patterns which are possible in this situation.

Task 1.

Pattern 1.1 "*States of the system with some possibility for normal operation*" (denoted as E_1^+).

Pattern 1.2: "*States of the system which guarantee an accident*" (denoted as E_1^-).

Task 2.

Pattern 2.1: "*Guaranteed reliable states of the system*" (denoted as E_2^+).

Pattern 2.2: "*Reliability of the states of the system is not guaranteed*" (denoted as E_2^-).

Our goal is to extract the way the system operates in the form of two discriminant Boolean functions f_2 and f_1. The first function is related to task 1, while the second function is related to task 2 (as defined above). Also observe that the following relations must be true: $E_1^+ \supset E_2^+$ and $f_1(\alpha) \geq f_2(\alpha)$ for all $\alpha \in E_n$, describing the system state, where $f_1(\alpha)$ and $f_2(\alpha)$ are the discriminant monotone Boolean functions for the first and second task, respectively.

Second illustrative example: Breast cancer diagnosis For the second illustrative example, consider the following nested classification problem related to breast cancer diagnosis. The first subproblem is related to the clinical question of whether a biopsy or short-term follow-up is necessary or not. The second subproblem is related to the question of whether the radiologist believes that the current case is highly suspicious for malignancy or not. It is assumed that if the radiologist believes that the case is malignant, then he/she will also definitely recommend a biopsy. More formally, these two subproblems are defined as follows:

The Clinical Management Subproblem: One and only one of the following two disjoint outcomes is possible:

1) "*Biopsy/short-term follow-up is necessary*," or:
2) "*Biopsy/short-term follow-up is not necessary*."

The Diagnosis Subproblem: Similarly as above, one and only one of the following two disjoint outcomes is possible. That is, a given case is:

1) "*Highly suspicious for malignancy*," or:
2) "*Not highly suspicious for malignancy.*"

It can be easily seen that the corresponding states satisfy nesting conditions similar to the ones in the first illustrative example.

Third illustrative example: Radioactivity contamination detection The third illustrative example is also diagnosis related. The issue now is how to perform radioactivity contamination tests. Usually, the more time demanding a test is, the more accurate the result will be. Therefore, an efficient strategy would be to perform the less time demanding tests first, and if need rises, to perform the more time demanding (and also more accurate) tests later (this is analogous to the previous breast cancer problem, in which case a biopsy can yield more accurate results but is also more expensive). The corresponding two nested subproblems can be defined as follows:

Subproblem 1: Diagnosis of radioactivity contamination, low-risk case.
Pattern 1: "*Necessarily contaminated.*"
Pattern 2: "*Not necessarily contaminated.*"
Subproblem 2: Very high risk for contamination:
Pattern 1: "*Contaminated and extra detection of contamination is necessary.*"
Pattern 2: "*Extra detection of contamination is not necessary.*"

Again, it can be easily seen that the corresponding states satisfy the nesting conditions of the first illustrative example. Next, the outline of an algorithm for solving this kind of problems is presented.

ALGORITHM A2:

Step 1: The user is asked to confirm the monotonicity for both tasks (i.e., the monotonicity of the functions underlying the patterns related to the previous two tasks or subproblems). The user is also required to confirm that the two functions are nested (i.e., $E_1^+ \supseteq E_2^+$ and $f_1(\alpha) \geq f_2(\alpha)$ for all $\alpha \in E_n$).

Step 2: If the user confirms that monotonicity should be present, then test for the monotonicity property in the initial examples for both tasks. Also, confirm that the available examples satisfy the "nesting" requirement.

If the user does not confirm monotonicity or the nesting requirement, then he must reformulate the problem characteristics (features) to reach monotonicity.

If monotonicity is not reached, then do not solve the problem with monotone Boolean functions and exit.

Step 3: Reject the examples which violate monotonicity. At this point, the following sets of examples are available: E_1^+, E_1^- and E_2^+, E_2^-, and $E_1^+ \supseteq E_2^+$ should hold.

Step 4: Use the algorithm in [12] (which is also described in Section 4.2) to create all Hansel chains for the number (denoted as n) of features in the current application.

Step 5: Use the initial (i.e., already classified) examples and the monotonicity property to infer all derivable classification values on the additional examples located in the above Hansel chains. For effective storage of the inferred examples, use the approach described in [23]. These ideas use the property of Hansel chains, according to which, at each iteration i, we need to store only two key examples from each Hansel chain. These two examples represent a maximal known upper zero and a minimal known lower unit for each Hansel chain.

Step 6: Generate two new candidate examples, say $\alpha_i^{(1)}$ and $\alpha_i^{(2)}$, to be considered for classification by oracles A_{f1} and A_{f2}, respectively. These examples are taken from Hansel chains according to the procedures described in Section 4.2. If no more examples are left, exit; the two functions have been fully restored. Otherwise, go to step 7.

Step 7: Given the above two examples $\alpha_i^{(1)}$ and $\alpha_i^{(2)}$, we send for classification the one which leads to the maximum number of additional examples to be classified (without new calls to the oracles) when the properties of monotonicity and nested functions are applied. How this is done is described in more detail next in steps 7.1 to 7.5.

Step 8: After either $\alpha_i^{(1)}$ and $\alpha_i^{(2)}$ has been classified by the appropriate oracle, use the monotonicity property to infer all derivable classification values of the additional examples located in the remaining Hansel chains. Go to step 6.

Next, we discuss step 7 in more detail. When step 7 is reached, there are two candidate examples, denoted as $\alpha_i^{(1)}$ and $\alpha_i^{(2)}$, which are considered for submission to the respective oracle. For the simultaneous restoration of f_1 and f_2, we need to send first for classification the example ($\alpha_i^{(1)}$ or $\alpha_i^{(2)}$) which best accelerates the learning process. Apparently, there are two possible alternatives: (i) first to ask oracle A_{f1} about the value of $f_1(\alpha_i^{(1)})$,

and (ii) first to ask the second oracle A_{f2} about the value of $f_2(\alpha_i^{(2)})$. These alternatives allow one to guarantee for both of oracles and functions f_1 and f_2 the optimum limits on the number of calls in (2.2). We select which example to submit first by computing two estimates N_1 and N_2 of the numbers of examples that can be classified if example $\alpha_i^{(1)}$ or $\alpha_i^{(2)}$ is classified first, respectively. Given the previous estimates, we decide which example to submit first for classification according to the following simple rules:

Rule 1: If $N_1 > N_2$, then ask A_{f1} about the $f_1(\alpha_i^{(1)})$ value.
Rule 2: If $N_1 < N_2$, then ask A_{f2} about the $f_2(\alpha_i^{(2)})$ value.
Rule 3: If $N_1 = N_2$, then choose $\alpha_i^{(1)}$ or $\alpha_i^{(2)}$ randomly.

Let us define N_1 and N_2 in more detail. Here, N_1 is an estimate of the number of elements $\alpha \in E_n$ such that $f_2(\alpha)$ becomes known without asking A_{f2}, if we know the value of $f_1(\alpha_i^{(1)})$. These $f_2(\alpha)$ values are inferred from the properties that, for all α, $f_2(\alpha) \leq f_1(\alpha)$, and the monotonicity of f_1 and f_2. In a similar manner, we define N_2 to estimate the number of $f_1(\alpha)$ values becoming known, if we know $f_2(\alpha_i^{(2)})$. Observe that there are two possible values, 0 and 1, for $f_1(\alpha_i^{(1)})$. First, temporarily assume that $f_1(\alpha_i^{(1)}) = 0$. Then, we can compute N_1^0 as the number of vectors $\alpha \in E_n$ for which the value of $f_2(\alpha)$ can be inferred without asking A_{f2}. Next, we assume that $f_1(\alpha_i^{(1)}) = 1$, and in a similar manner, we compute the quantity N_1^1. When the previous two quantities are computed, then the value of N_1 can be estimated as the sum of N_1^0 and N_1^1. That is, we use $N_1 = N_1^0 + N_1^1$.

In a similar manner, the quantities N_2^0 and N_2^1 can be defined, and then N_2 can be estimated as their sum. That is, $N_2 = N_2^0 + N_2^1$. At this point, please observe that the computation of the above numbers can be simplified as follows. We can just compute the number of these elements on the Hansel chains of the current and next length. For all other elements, this extension is not needed. It will be done later when we come to these elements. This follows from the proof of the Hansel theorem and lemma [14].

Based on the above definitions and discussions, we propose to use the following algorithm (described as steps 7.1 to 7.6) in order to decide which example to choose.

Step 7.1: Compute (N_1^0) by first assuming that $f_1(\alpha_i^{(1)}) = 0$.
Step 7.2: Compute (N_1^1) by first assuming that $f_1(\alpha_i^{(1)}) = 1$.
Step 7.3: Compute (N_2^0) by first assuming that $f_2(\alpha_i^{(2)}) = 0$.
Step 7.4: Compute (N_2^1) by first assuming that $f_2(\alpha_i^{(2)}) = 1$.
Step 7.5: Compute N_1 and N_2 as the sum of the appropriate pair of the previous quantities.

Step 7.6: If $N_1 > N_2$, then ask A_{f1} about $f_1(\alpha_i^{(1)})$.
If $N_1 < N_2$, then ask A_{f2} about $f_2(\alpha_i^{(2)})$.
If $N_1 = N_2$, then randomly submit $\alpha_i^{(1)}$ or $\alpha_i^{(2)}$ to the appropriate oracle.

Comment. In step 7, we realized a myopic algorithm, which can be extended. Recall that we considered the possibilities of classification values $f_1(\alpha_i^{(1)})$ and $f_2(\alpha_i^{(2)})$ at the next level (denoted as the ith level). One may wish to extend this to the $(i=1)$ level and so on. In this way, the estimates will be more accurate (this is similar to playing chess by considering two, three, or more moves in advance). However, although this extension of Algorithm A2 may decrease the number of calls, naturally it will lead to an increase in computation time. Therefore, we propose to use the depth (i.e., the number of look-ahead levels) of this search as a parameter which is application-dependent.

4. AN ALGORITHM FOR RESTORING A MONOTONE BOOLEAN FUNCTION

4.1. GENERAL SCHEME OF THE ALGORITHM

Next, we present algorithm RESTORE for the interactive restoration of a monotone Boolean function, and two procedures GENERATE and EXPAND in a pseudoprogramming language.

ALGORITHM "RESTORE"

Input. Dimension n of the binary space and access to an oracle A_f.

Output. A monotone Boolean function restored after a minimal number (according to Shannon criterion as it is given as relation (2.2) of calls to the oracle A_f.

Method.

1) Construction of Hansel chains (see Section 4.2)
2) Restoration of a monotone Boolean function starting from chains of minimal length and finishing with chains of maximal length. This ensures that the number of membership calls to the oracle A_f is no more than the limit presented in formula (2.2), which guarantees the number of calls to an oracle A_f to be no more than the limit presented in formula (2.1).

BEGIN;
Set $i=1$; {initialization}
DO WHILE (function $f(x)$ is not entirely restored)
Step 1. Use procedure GENERATE to generate element α_i;
Step 2. Call oracle A_f to retrieve the value of $f(\alpha_i)$;

Step 3. Use procedure EXPAND to deduce the values of other elements in Hansel chains (i.e., sequences of examples in E_n) by using the value of $f(\alpha_i)$, the structure of element α_i, and the monotonicity property.

Step 3. Set $i \leftarrow i+1$;

RETURN

END;

PROCEDURE "GENERATE": Used to generate an element α_i to be classified by the oracle A_f.

Input. Dimension n of the binary space.

Output. The next element α_i to send for classification by the oracle A_f.

Method: Proceeding from minimal to maximal Hansel chains.

IF $i=1$ THEN {where i is the index of the current element}

BEGIN

Step 1.1. Retrieve all Hansel chains of minimal length;

Step 1.2. Randomly choose the first chain C_1 among the chains retrieved in step 1.1;

Step 1.3. Set the first element α_1 as the minimal element of chain C_1;

END

ELSE

BEGIN

Set $k=1$ {where k is the index number of a Hansel chain};

DO WHILE (NOT all Hansel chains are tested)

Step 2.1. Find the largest element α_i in chain C_k, which still has no confirmed $f(\alpha_i)$ value;

Step 2.2. If step 2.1 did not return an element α_i, then randomly select the next Hansel chain C_{k+1} of the same length as the one of the current chain C_k;

Step 2.3. Find the least element α_i from chain C_{k+1}, which still has no confirmed $f(\alpha_i)$ value;

Step 2.4. If Step 2.3 did not return an element α_i, then randomly choose chain C_{k+1} of the next available length;

Step 2.5 Set $k \leftarrow k+1$;

RETURN

END

PROCEDURE "EXPAND": Used to expand the $f(a_i)$ value for other elements using monotonicity and Hansel chains.

Input. The $f(a_i)$ value.

Output. An extended set of elements with known $f(x)$ values.

Method: The method is based on the properties: if $x \geq \alpha_i$ and $f(\alpha_i)=1$, then $f(x)=1$; and if $x \leq \alpha_i$ and $f(\alpha_i)=0$, then $f(x)=0$.

BEGIN
Step 1. Obtain x such that $x \geq \alpha_i$ or $x \leq \alpha_i$ and x is in a chain of the length l or $l+2$.
Step 2. If $f(\alpha_i)=1$, then $\forall x\ (x \geq \alpha_i)$ set $f(x)=1$;
If $f(\alpha_i)=0$, then $\forall x\ (x \leq \alpha_i)$ set $f(x)=0$;
Step 3. Store the $f(x)$ values which were obtained in step 2;
END

Comment. To obtain x in step 1, we generate vectors $\{x\}$ only from Hansel chains of current and the next length. This follows from the Hansel lemma [14].

4.2. CONSTRUCTION OF HANSEL CHAINS

Several steps in the previous algorithms deal with Hansel chains. Next we describe how to construct all Hansel chains for a given state space E_n of dimension n. First, we give a formal definition of a general chain.

A *chain* is a sequence of binary vectors (examples) $\alpha_1, \alpha_2, \ldots, \alpha_i, \alpha_{i+1}, \ldots, \alpha_q$, such that α_{i+1} is obtained from a_i by changing a "0" element to "1." That is, there is an index k such that $\alpha_{i,k}=0$, $\alpha_{i+1,k}=1$, and for any $t \neq k$ the following is true: $\alpha_{i,t}=\alpha_{i+1,t}$. For instance, the list $\langle 01000, 01100, 01110\rangle$ of three vectors is a chain.

To construct all Hansel chains, we will use an iterative procedure as follows: Let $E_n=\{0,1\}^n$ be the n-dimensional binary cube. All chains for E_n are constructed from chains for E_{n-1}. Therefore, we begin the construction for E_n by starting with E_1 and iteratively proceeding to E_n.

(i) Chains for E_1.

For E_1, there is only a single (trivial) chain and it is $\langle 0,1\rangle$.

(ii) Chains for E_2.

We take E_1 and add at the beginning of each one of its chains the element (0). Thus, we obtain the set $\{00,01\}$. This set is called $E_2^{\min}$. Similarly, by adding "1" to $E_1=\{(0),(1)\}$, we construct the set $E_2^{\max}=\{10,11\}$. To simplify notation, often we will omit "( )" for vectors as (10)

and (11). Both $E_2^{\min}$ and $E_2^{\max}$ are isomorphic to E_1, because they have the isomorphic chains $\langle 00,01\rangle$ and $\langle 10,11\rangle$. The union of $E_2^{\min}$ and $E_2^{\max}$ is E_2. That is, $E_2 = E_2^{\min} \cup E_2^{\max}$. Observe that the chains $\langle 00,01\rangle$ and $\langle 10,11\rangle$ cover entirely E_2; however, they are not Hansel chains. To obtain Hansel chains, we need to modify them as follows.

Adjust the chain $\langle 00,01\rangle$ by adding the maximum element (11) from the chain $\langle 10,11\rangle$. That is, obtain a new chain $\langle 00,01,11\rangle$. Also reject element (11) from the chain $\langle 10,11\rangle$. Hence, we obtain the two new chains $\langle 00,01,11\rangle$ and $\langle 10\rangle$. These chains are the *Hansel chains* [14] for E_2. That is, $E_2 = \{\langle 00,01,11\rangle, \langle 10\rangle\}$.

(iii) Chains for E_3.

The Hansel chains for E_3 are constructed in a manner similar to the chains for E_2. First, we double and adjust the Hansel chains of E_2 to obtain $E_3^{\min}$ and $E_3^{\max}$. The following relations is also true:

$$E_3 = E_3^{\min} \cup E_3^{\max},$$

where $E_3^{\min} = \{\langle 000,001,011\rangle, \langle 010\rangle\}$ and $E_3^{\max} = \{\langle 100,101,111\rangle, \langle 110\rangle\}$. We do the same chain modification as for E_2. That is, first we choose two isomorphic chains. At first, let it be two maximal-length chains $\langle 000,001,011\rangle$ and $\langle 100,101,111\rangle$. We add the maximal element (111) form $\langle 100,101,111\rangle$ to $\langle 000,001,011\rangle$ and drop it from $\langle 100,101,111\rangle$. In this way, we obtain the two new chains $\langle 000,001,011,111\rangle$ and $\langle 100,101\rangle$. Next, we repeat this procedure for the rest of the isomorphic chains $\langle 010\rangle$ and $\langle 110\rangle$. In this simple case, we will have just one new chain $\langle 010,110\rangle$ (note that the second chain will be empty). Therefore, E_3 consists of the three Hansel chains $\langle 010,110\rangle$, $\langle 100,101\rangle$, and $\langle 000,001,011,111\rangle$. That is, the following is true:

$$E_3 = \{\langle 010,110\rangle, \langle 100,101\rangle, \langle 000,001,011,111\rangle\}.$$

In a similar manner, one can construct the Hansel chains for E_4, E_5, and so on. Below, we present a general scheme for E_n under the assumption that we already have all Hansel chains for E_{n-1}. Note that the Hansel chains of E_{n-1} can be obtained recursively from the Hansel chains of $E_1, E_2, \ldots, E_{n-2}$.

(iv) Chains for E_n.

Suppose that $E_{n-1}=\{C_1,\ldots,C_i,\ldots,C_k\}$, where each C_i is a Hansel chain for E_{n-1}. First, we double E_{n-1}, and by using the "0" and "1" elements as before, we construct the two isomorphic chains:

$$E_n^{\min}=\{0C_1,0C_2,\ldots,0C_k\},$$

and

$$E_n^{\max}=\{1C_1,1C_2,\ldots,1C_k\}.$$

The sets $E_n^{\min}$ and $E_n^{\max}$ entirely cover E_n. That is, the following is true:

$$E_n=E_n^{\min}\cup E_n^{\max}.$$

Next, let us consider the chain $C_i=\{c_{i1},\ldots,c_{ij}\ldots,c_{im(i)}\}$ and the pair of the isomorphic Hansel chains $0C_i$ and $1C_i$, in which $0C_i=\{0c_{i1},\ldots,0c_{ij},\ldots,0c_{im(i)}\}$ and $1C_i=\{1c_{i1},\ldots,1c_{ij}\ldots,1c_{im(i)}\}$. Note that in the previous expressions, $0c_{ij}$ and $1c_{ij}$ are *n-dimensional binary vectors*, while c_{ij} is an $n-1$*-dimensional binary vector*. We construct a new pair of Hansel chains $0C_i^H=\{0c_{i1},\ldots,0c_{ij},\ldots,0c_{im(i)},1c_{im(i)}\}$ and $1C_i^H=\{1c_{i1},\ldots,1c_{ij},\ldots,1c_{im(i)-1}\}$ from $0C_i=\{0c_{i1},\ldots,0c_{ij},\ldots,0c_{im(i)}\}$ and $1C_i=\{1c_{i1},\ldots,1c_{ij},\ldots,1c_{im(i)}\}$ by using the same "*cut and add*" procedure as was the case for E_3. We repeat this operation for all pairs of isomorphic chains in $E_n^{\min}$ and $E_n^{\max}$ to obtain the entire set of Hansel chains for E_n.

Next, in Figures 1 and 2, we illustrate this procedure of chain construction for E_2 and E_3. Figure 1 shows the transformation of E_1 in $E_2^{\max}$ and $E_2^{\min}$, and after that, the construction of the new chains from them. The nodes of a chain are connected with solid lines. At first we have an edge

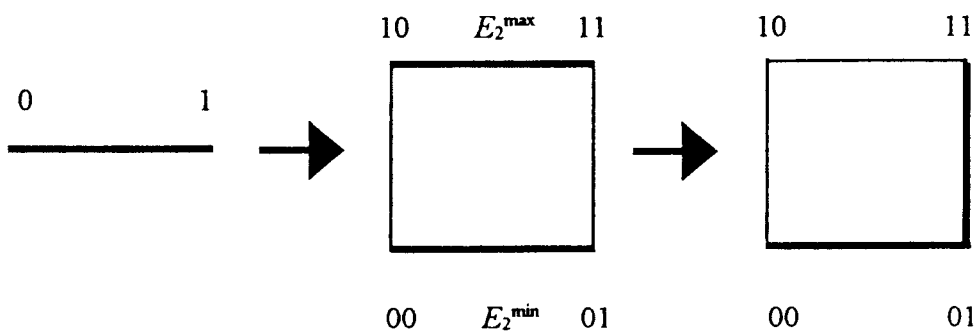

Fig. 1. Construction of Hansel chains for E_2.

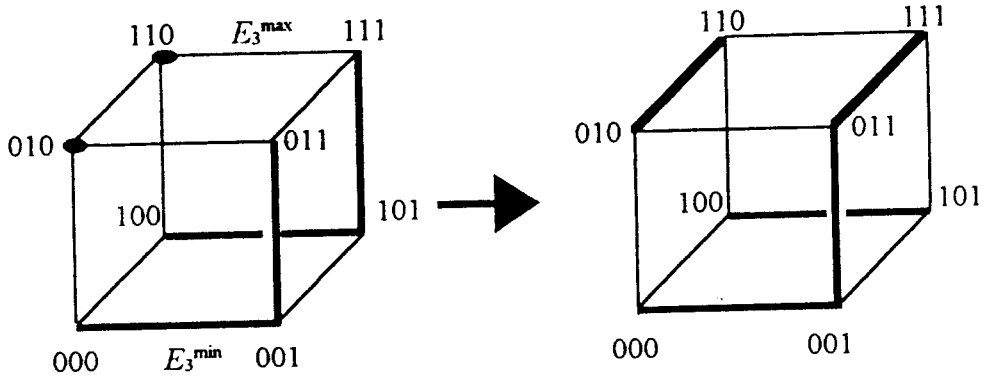

Fig. 2. Construction of Hansel chains for E_3.

between (10) and (11). Next we cut it and obtain a new chain $\langle 00,01,11\rangle$ by combining (11) to the chain $\langle 00,01\rangle$. The node (10) forms a trivial chain with a single element. Figure 2 begins from two copies of the output chain shown in Figure 1 with an added "0" to the first of them and a "1" to the second one. The output in Figure 2 presents the chains in E_3, which are also connected with solid lines. These chains were obtained with the same "*cut and add*" procedure.

The justification of the above construction of this section is the proof of the Hansel lemma. This lemma uses the concepts of a *complement* element, an *interval*, and a *square*. Three binary vectors $\alpha_1, \alpha_2, \alpha_3$ (where $\alpha_1 \prec \alpha_2 \prec \alpha_3$) form an *interval* $[\alpha_1, \alpha_3]$, if there are no other vectors between (α_1, α_2) and (α_2, α_3). The vector β is a *complement* of α_2 in the interval $[\alpha_1, \alpha_3]$ if $\alpha_1 \prec \beta \prec \alpha_3$. In this case, we also say that the vectors α_1, α_2, α_3 and β form a *square* (see Fig. 3). An instance of a square is $\alpha_1 = (01000)$, $\alpha_2 = (01100)$, $\alpha_3 = (01110)$, and $\beta = (01010)$. The Hansel lemma is stated as follows.

LEMMA (Hansel, [14]). *The binary cube E_n is covered by $\binom{n}{\lfloor n/2 \rfloor}$ disjoint chains which have the following properties*:

(*a*) *The number of chains of length $n-2p+1$ is $\binom{n}{p} - \binom{n}{p-1}$ (where $0 \le p \le \lfloor n/2 \rfloor$).*

(*b*) *The minimal element of any of the chains of length $n-2p+1$ is a vector with p units* ("1") *and $n-p$ zeros* ("0").

(*c*) *For any three sequential elements $\alpha_1 \prec \alpha_2 \prec \alpha_3$ from a chain, the complement to α_2 in the interval $[\alpha_1, \alpha_3]$ belongs to a chain of length $n-2p-1$.*

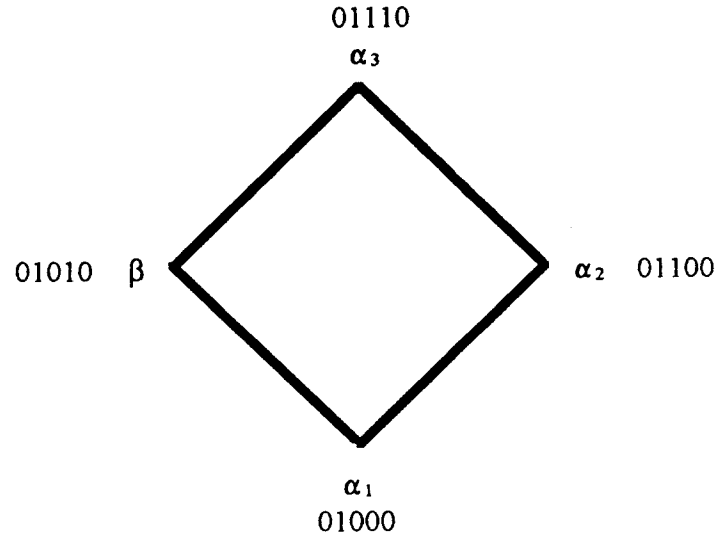

Fig. 3. A binary square.

5. A COMPUTATIONAL INTERACTIVE HIERARCHICAL EXPERIMENT

5.1. PROBLEM DESCRIPTION

In this section, we demonstrate the previous issues in terms of an illustrative example from the area of breast cancer diagnosis. It is assumed that some preliminary medical tests have been performed (such as a mammogram) and the medical doctor wishes to determine what to do next. That is, should he/she proceed with a biopsy or not? The problem considered in this section is a *nested* one because it is comprised of two interrelated subproblems. The first problem is related to the question of whether a biopsy is necessary. The second subproblem is related to the question of whether the human expert (medical doctor) believes that the current case represents a malignant case or not. It is assumed that if the doctor believes that the case is malignant, then he/she will also definitely recommend a biopsy. Recall that in Section 3 we defined these two subproblems as follows:

The Clinical Management Subproblem: One and only one of the following two disjoint outcomes is possible:

1) "*Biopsy/short-term follow-up is necessary*," or:
2) "*Biopsy/short-term follow-up is not necessary*."

The Diagnosis Subproblem: Similarly as above, one and only one of the following two disjoint outcomes is possible. That is, a given case is:

1) "*Highly suspicious for malignancy*," or:
2) "*Not highly suspicious for malignancy.*"

The previous two interrelated subproblems can be formulated as the restoration problem of two nested monotone Boolean functions. A medical expert was explained the concept of monotonicity and he felt comfortable with the idea of using monotone Boolean functions. Moreover, the dialogue which followed confirmed the validity of this assumption. The underlying Boolean functions were defined on 11 binary atoms. This simplification was done to illustrate the applicability of the proposed methodology. More details on this problem can be found in [25].

From the above discussion, it follows that now we will work in the 11-dimensional binary space (i.e., in E_{11}). The two corresponding functions are f_1 and f_2. Function f_1 returns true (1) value if the decision is "*biopsy/short-term follow-up is necessary*," false (0) otherwise. Similarly, function f_2 returns true (1) value if the decision is "*highly suspicious for malignancy*," false (0) otherwise.

Full restoration of either one of the functions f_i (for $i = 1,2$) without any optimization on the dialogue process would have required up to $2^{11} = 2048$ calls (membership inquires) to an expert (i.e., the A_f operator or medically oriented binary attributes). However, according to the Hansel lemma (i.e., relation (2.2) and under the assumption of monotony, an optimal (i.e., with a minimal number of accesses) dialogue for restoring a monotone Boolean would require at most $\binom{11}{5} + \binom{11}{6} = 2 \times 462 = 924$ calls to the oracle. Please observe that this new value is 2.36 times smaller than the pervious upper limit of 2048 calls. However, even this upper limit of 924 calls can be reduced further, as is explained in the next section.

5.2. *HIERARCHICAL DECOMPOSITION*

In this particular breast cancer diagnosis problem, the medical expert indicated that the original 11 binary attributes can be organized in terms of a *hierarchy*. This hierarchy follows from the definition of the 11 medically oriented binary attributes. Next, it will be shown that this hierarchy can be exploited to reduce the required number of calls to the "oracle" even further. We will not describe the medical justification of this hierarchy here (more details can be found in [25]). The hierarchy used in

this example is given as follow:

Level 1 (5 attributes)		Level 2 (all 11 attributes)
x_1	$\leftarrow$	w_1, w_2, w_3
x_2	$\leftarrow$	y_1, y_2, y_3, y_4, y_5
x_3	$\leftarrow$	x_3
x_4	$\leftarrow$	x_4
x_5	$\leftarrow$	x_5

In this hierarchy, the two attributes x_1 and x_2 correspond to two monotone Boolean functions which also have to be restored. That is, the following is true:

$$x_1 = \varphi(w_1, w_2, w_3),$$

and

$$x_2 = \psi(y_1, y_2, y_3, y_4, y_5).$$

The expert indicated that these two functions φ and ψ should be common to both problems (i.e., the biopsy and cancer problems). Therefore, the following relation is true regarding the f_i (for $i = 1,2$) and the two φ and ψ functions:

$$f_i(x_1, x_2, x_3, x_4, x_5) = f_i(\varphi(w_1, w_2, w_3), \psi(y_1, y_2, y_3, y_4, y_5), x_3, x_4, x_5)).$$

Given the above analysis and decomposition, then according to the Hansel theorem (i.e., relation (2.2), it follows that for the restoration problem at level 1, it would have been required to have at most $\binom{5}{2} + \binom{5}{2} = 20$ calls to the "oracle." Observe that without the exploitation of the monotonicity property, an exhaustive dialogue requires $2^5 = 32$ calls, which is 1.6 times higher. Also, the exploitation of a hierarchy can even benefit the restoration of general (i.e., not necessarily monotone) Boolean functions.

The above bounds represent upper limits. When we actually interviewed the medical expert, the target function f_1 was fully restored with only 13 membership inquires (calls). Therefore, the number of the actual calls was

1.5 times less than the upper limit given by Hansel's theorem and 2.5 times less than the exhaustive search (which would have required up to 32 calls). Also, the same 13 calls were adequate to restore each of $f_2(x_1, x_2, x_3, x_4, x_5)$ and $x_2 = \psi(y_1, y_2, y_3, y_4, y_5)$ functions.

Next, x_1 was restored as a function $\varphi(w_1, w_2, w_3)$. In this case, the exhaustive search would have required $2^3 = 8$ calls. On the other hand, Hansel's lemma provides an upper limit of $\binom{1}{3} + \binom{2}{3} = 6$ calls. This value is 1.33 times less than what the exhaustive search would have required. Although the previous numbers of calls are not excessively high, they still provide a sense of the potential of significant savings which can be achieved if monotonicity is established and utilized. The numbers in this example are supposed only to be for illustrative purposes in clarifying the proposed methodology.

In the following paragraphs, we present the optimal dialogue restoring the $\varphi(w_1, w_2, w_3)$ function. Recall from Section 4.2 that the Hansel chains for the three-dimensional space E_3 are $\{\langle 010, 111\rangle, \langle 100, 101\rangle, \langle 000, 001, 011, 111\rangle\}$. The medical expert was asked to evaluate the value of some of the above examples. Given the value of certain examples in these chains and using the monotonicity property, we were able to induce the values of more examples. The results are summarized below:

	Chain 1		Chain 2		Chain 3			
Vectors	$\langle 010,$	$110\rangle$	$\langle 100,$	$101\rangle$	$\langle 000,$	$001,$	$011,$	$111\rangle$
Value of φ	1^*	1	0^*	1^*	0	0^*	1	1

In the above illustration, "1*" and "0*" indicate answers given directly by the expert. For instance, during the dialogue, the expert decided that $\varphi(010) = 1$, $\varphi(101) = 1$, and $\varphi(100) = \varphi(001) = 0$. The values without an asterisk (i.e., when we only have "1" or "0") were obtained by utilizing the monotonicity property.

In this experiment, the four calls about the function values for the examples (010), (100), (101), and (001) were sufficient to lead to full restoration. Monotonicity allowed us to extend the previous four values to infer the values of the rest of the examples. For instance, from the expert's testimony that $\varphi(010) = 1$, it is derived that $\varphi(110) = 1$, and from $\varphi(001) = 0$, it is derived that $\varphi(000) = 0$. Observe that these four calls to the expert

are 2 times less than the maximum number of calls (i.e., 8) and 1.5 times less than the guaranteed number of calls provided by Hansel's theorem (which is equal to 6).

In order to describe function $\varphi(w_1, w_2, w_3)$ in the standard disjunctive normal form (DNF), we should include from each one of the chains 1–3 above their minimal upper units, i.e., (010), (101), and (011) presented as w_2, w_1w_3, w_2w_3, respectively. As a result, we obtained their DNF expression. This DNF, after simplification, is $w_2 \vee w_1w_3$. That is, the following is true:

$$x_1 = \varphi(w_1, w_2, w_3) = w_2 \vee w_1w_3 \vee w_2w_3 = w_2 \vee w_1w_3. \tag{5.1}$$

In a similar manner, we can obtain Boolean expressions for $x_2 = \psi(y_1, y_2, y_3, y_4, y_5)$ from the Hansel chains for E_5:

$$x_2 = \psi(y_1, y_2, y_3, y_4, y_5) = y_2 \vee y_1 \vee y_3y_4y_5. \tag{5.2}$$

Similarly for level 1, we obtained the target functions of x_1, x_2, x_3, x_4, x_5 for the **biopsy** subproblem to be defined as follows:

$$f_1(x) = x_2x_4 \vee x_1x_2 \vee x_1x_4 \vee x_3 \vee x_5, \tag{5.3}$$

and for the **cancer** subproblem to be defined as

$$f_2(x) = x_1x_2 \vee x_3 \vee x_2x_5 \vee x_1x_5 \vee x_4x_5 = x_1x_2 \vee x_3 \vee (x_2 \vee x_1 \vee x_4)x_5. \tag{5.4}$$

The above expression allows one to compare the two functions f_1 and f_2. Observe that $f_1(x) = A \vee (x_2 \vee x_1)x_4 \vee x_5$, and $f_2(x) = A \vee (x_2 \vee x_1 \vee x_4)x_5$, where $A = x_1x_2 \vee x_3$. Hence, these two functions differ only in the parts $(x_2 \vee x_1)x_4 \vee x_5$ and $(x_2 \vee x_1 \vee x_4)x_5$, which may be important information for additional medical analysis. Next we use expressions (5.1) and (5.2) in (5.3) and (5.4) to define the explicit form f_1 and f_2 as the proposed final formulas for performing **biopsy**:

$$\begin{aligned} f_1(x) &= x_2x_4 \vee x_1x_2 \vee x_1x_4 \vee x_3 \vee x_5 \\ &= (y_2 \vee y_1 \vee y_3y_4y_5)x_4 \vee (w_2 \vee w_1w_3)(y_2 \vee y_1 \vee y_3y_4y_5) \\ &\quad \vee (w_2 \vee w_1w_3)x_4 \vee x_3 \vee x_5. \end{aligned} \tag{5.5}$$

and for **cancer**:

$$f_2(x) = x_1x_2 \vee x_3 \vee x_2x_5 \vee x_1x_5 \vee x_4x_5$$
$$= (w_2 \vee w_1w_3)(y_2 \vee y_1 \vee y_3y_4y_5) \vee x_3 \vee (y_2 \vee y_1 \vee y_3y_4y_5)x_5$$
$$\vee (w_2 \vee w_1w_3)x_5 \vee x_4x_5. \tag{5.6}$$

In total, we needed *29 calls* to restore f_1 in the following form:

$$f_i(x_1, x_2, x_3, x_4, x_5)$$
$$= f_i(\varphi(w_1, w_2, w_3), \psi(y_1, y_2, y_3, y_4, y_5), x_2, x_3, x_4, x_5)).$$

The same 29 calls were also needed to restore f_2 independently. In reality, however, we used 42 calls instead of 58 calls to restore *both functions* due to the fact that the component functions φ and ψ are the same for the target functions of both problems. For the record, we spent for the dialogue with the medical expert no more than one hour.

At this point it is interesting to observe that if one wishes to restore the *non*monotone Boolean function which corresponds to the concept: "*biopsy and not cancer*, then this can be achieved easily as follows. First note that the above concept represents cases in which surgery can potentially be avoided. This concept can be presented with the composite formula $f_1 \& \rceil f_2$, and thus it can be computed by using expressions (5.5) and (5.6). Therefore, a total number of 42 calls is just sufficient to restore this function (which is both nonmonotone and very complex mathematically).

5.3. *INDEPENDENT SEARCH OF FUNCTIONS*

Below, we systematically present how we obtained the results in Section 5.2. The main technical tool is Table 1. This table represents the dialogue which was executed based on the algorithm for problem 1 (in Section 3). The information in Table 1 resulted in the formation of formulas (5.2) to (5.6). The first column of Table 1 indicates examples: the first number indicates a Hansel chain and the second number indicates the location of the vector within that chain. The second column presents binary vectors. The third, fourth, and fifth columns present values of the functions f_1, f_2, and ψ, respectively. Asterisks "*" show values obtained directly from the expert. Values without asterisks were inferred by using the previous values and the property of monotonicity.

TABLE 1
Dialogue Sequences for Numerical Experiment

#	Vector	f_1	f_2	ψ	Extension $1 \to 1$	Extension $0 \to 0$	$\tilde{f_1}$	$\tilde{f_2}$	$\hat{f_1}$	$\hat{f_2}$
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
Chain 1										
1.1	(01100)	1*	1*	1*	1.2; 6.3; 7.3	7.1; 8.1	1*	1*	1	1*
1.2	(11100)	1	1	1	6.4; 7.4	5.1; 3.1	1	1	1	1
Chain 2										
2.1	(01010)	1*	0*	1*	2.2; 6.3; 8.3	6.1; 8.1	1*	0*	1*	0*
2.2	(11010)	1	1*	1	6.4; 8.4	3.1; 6.1	1	1*	1	1*
Chain 3										
3.1	(11000)	1*	1*	1*	3.2	8.1; 9.1	1	1*	1	1*
3.2	(11001)	1	1	1	7.4; 8.4	8.2; 9.2	1	1	1	1
Chain 4										
4.1	(10010)	1*	0*	1*	4.2; 9.3	6.1; 9.1	1*	0*	1*	0*
4.2	(10110)	1	1*	1	6.4; 9.4	6.2; 5.1	1	1*	1	1*
Chain 5										
5.1	(10100)	1*	1*	1*	5.2	7.1; 9.1	1	1*	1	1*
5.2	(10101)	1	1	1	7.4;9 .4	7.2; 9.2	1	1	1	1
Chain 6										
6.1	(00010)	0*	0	0*	6.2; 10.3	10.1	0*	0	0*	0*
6.2	(00110)	1*	1*	0*	6.3; 10.4	7.1	1	1*	1*	1*
6.3	(01110)	1	1	1	6.4; 10.5		1	1	1	1
6.4	(11110)	1	1	1	10.6		1	1	1	1
Chain 7										
7.1	(00100)	1*	1*	0*	7.2; 10.4	10.1	1	1*	1	1*
7.2	(00101)	1	1	0*	7.3; 10.4	10.2	1	1	1	1
7.3	(01101)	1	1	1*	7.4; 10.5	8.2; 10.2	1	1	1	1
7.4	(11101)	1	1	1	5.6		1	1	1	1
Chain 8										
8.1	(01000)	0*	0	1*	8.2	10.1	0*	0	0*	0
8.2	(01001)	1*	1*	1	8.3	10.2	1	1*	1	1*
8.3	(01011)	1	1	1	8.4	10.3	1	1	1	1
8.4	(11011)	1	1	1	10.6	9.3	1	1	1	1
Chain 9										
9.1	(10000)	0*	0	1*	9.2	10.1	0*	0	0*	0*
9.2	(10001)	1*	1*	1	9.3	10.2	1	1*	1*	1*
9.3	(10011)	1	1	1	9.4	10.3	1	1	1	1
9.4	(10111)	1	1	1	10.6	10.4	1	1	1	1
Chain 10										
10.1	(00000)	0	0	0	10.2		0	0	0	0
10.2	(00001)	1*	0*	0	10.3		1*	0*	1*	0
10.3	(00011)	1	1*	0	10.4		1	1*	1	1*
10.4	(00111)	1	1	1	10.5		1	1	1	1
10.5	(01111)	1	1	1	10.6		1	1	1	1
10.6	(11111)	1	1	1			1	1	1	1
Total Calls		**13**	**13**	**12**			**7**	**13**	**22**	

The sixth column shows the indexes of vectors for which we can use monotonicity if a given function evaluates to 1 (i.e., true value). For instance, for vector #7.1 we have $f_1(00100) = 1^*$ (as given by the expert); hence, we can also set (by utilizing the monotonicity property) $f_1(00101) = 1$, because vector (00101) is vector #7.2. We can also do the same with vector #10.4.

Similarly, the seventh column represents the indexes of the vectors for which we can use monotonicity if a given function evaluates to 0 (i.e., false value). For instance, for the vector #6.1 we have $f_1(00010) = 0^*$ (as given by the expert) and thus we can infer (by utilizing monotonicity) that $f_1(00000) = 0$, where (00000) is vector #10.1. The values in column 3 for f_1 are derived by *up-down sliding* in Table 1 according to the following five steps:

Step 1. Begin from the first vector #1.1 (i.e., (01100)).
Action: Ask the expert about the value of $f_1(01100)$.
Result: The expert reported that $f_1(01100) = 1$.

Step 2. Write "1*" next to vector #1.1 (and under column 3). Recall that an asterisk denotes an answer directly provided by the expert. Apparently, if the reply were false (0), then we had to write "0*" in column 3. The case of having a false value corresponds to column 7 in Table 1.

Step 3. **Action:** Go to column 6 to find vectors to extend the true (1) value.
Result: Vectors #1.2, #6.3, and #7.3.
Comment: They should yield (because of monotonicity) true value (1).
Therefore, in column 3 and next to examples #1.2, #6.3, and #7.3, the values of the function must be (1) (note that now no asterisk is used).

Step 4. **Action:** Go to vector #1.2 (next vector while sliding down). Check whether the value of $f_1(11100)$ has already been fixed or not.
If the value of $f_1(11100)$ is not fixed (i.e., it is empty), then repeat steps 1–3, above, for this new vector (i.e., vector #1.2).
If $f_1(11100)$ is not empty (i.e., it has been already fixed), then go to the next vector while sliding down (i.e., move to vector #2.1).
(Note that if the value has not been fixed yet, then we will denote it by $f_i(x) = e$; for *e*mpty.)
Result: $f_1(11100) \neq e$. Extend the values of the function for the vectors #6.4 and #7.4. and go to the next vector in the table (i.e., move to vector #2.1).

Step 5. The next vector is #2.1. Continue as above with step 4, but now we have vector #2.1 (instead of vector #1.2).

The above procedure is repeated until all vectors and functions have been covered. The interpretation of what happens in columns 8 to 11 is provided in the next subsection. In order to construct formula (5.3) (which was shown in Section 5.2), one needs to concentrate on the information depicted in columns 2 and 3 in Table 1. One needs to take the first vector marked with "1*" in each one of the chains and construct for each of these vectors a conjunction of nonzero components. For instance, for the vector (01010) in chain 2 the corresponding conjunction is x_2x_4. Similarly, from chain 6 we have taken the "1" components in the vector (00110) and formed the conjunction x_3x_4.

We obtained the Boolean expressions (5.2) for $x_2 = \psi(y_1, y_2, y_3, y_4, y_5)$ from the information depicted in Table 1 (columns 2 and 5) with the following steps: (i) First find all the maximal lower units for all chains as elementary conjunctions; (ii) exclude the redundant terms (conjunctions) from the end formula. Thus, at first from Table 1 (columns 2,5) we obtained $x_2 = y_1y_2 \vee y_2y_3 \vee y_2y_4 \vee y_1y_3 \vee y_1y_4 \vee y_2y_3y_4 \vee y_2y_3y_5 \vee y_2 \vee y_1 \vee y_3y_4y_5$, and then we simplified it to $y_2 \vee y_1 \vee y_3y_4y_5$.

Similarly as above, from columns (2,3,4) in Table 1, we obtained the initial components of the target functions of x_1, x_2, x_3, x_4, x_5 for the **bipsy** subproblem as follows:

$$f_1(x) = x_2x_3 \vee x_2x_4 \vee x_1x_2 \vee x_1x_4 \vee x_1x_3 \vee x_3x_4 \vee x_3 \vee x_2x_5 \vee x_1x_5 \vee x_5,$$

and for the **cancer** subproblem to be defined as:

$$f_2(x) = x_2x_3 \vee x_1x_2x_4 \vee x_1x_2 \vee x_1x_3x_4 \vee x_1x_3 \vee x_3x_4 \vee x_3 \vee x_2x_5 \vee x_1x_5 \vee x_4x_5.$$

The simplification of these disjunctive normal form (DNF) expressions allowed us to exclude some redundant conjunctions. For instance, in x_2 the term y_1y_4 is not necessary, because y_1 covers it. Thus, the right-hand side parts in expressions (5.1) to (5.4) form the minimal DNFs.

5.4. SEQUENTIAL SEARCH FOR NESTED FUNCTIONS

Next, we show how it is possible to further decrease the number of calls using a modification of the algorithm described for solving Problem 2. Recall that the functions f_1 and f_2 are nested because $E_2^+ \subseteq E_1^+$. That is, for any input example x, the following is true: $f_2(x) \leq f_1(x)$. The last statement means that if we recognize cancer, then we recognize the necessity of biopsy, too. The property $f_2(x) \leq f_1(x)$ means that

$$\text{if } f_2(x) = 1, \quad \text{then } f_1(x) = 1, \tag{5.7}$$

and

$$\text{if } f_1(x) = 0, \quad \text{then } f_2(x) = 0. \tag{5.8}$$

The above realization permits one to avoid calls to the expert for determining the right-hand sides of (5.7) and (5.8), if one knows the values of the left-hand side.

The above idea can be used to make the dialogue with the expert more efficient. This was done in this experiment and is described in Table 1. Column 8 in Table 1 shows the results of using the values of f_2 (column 4) and property (5.7) to restore function f_1. For instance, for vector #6.2, according to the expert we have $f_2(00110) = 1$. Hence, by using (5.7) we should also have $f_1(00110) = 1$ *without* an additional call to the expert. Column 9 represents the same situation for f_2, if one knows the expert's answers for f_1. In this case, we use the pair {relation (5.8), column (4)} instead of the pair {relation (5.7), column 5}, as was the case before.

Note that the asterisks "*" in columns 8 and 9 show the necessarily needed calls. In order to restore function f_1 by using information regarding values of f_2, we asked the expert 7 times, instead of the 13 calls we had to use for the independent restoration of function f_1. That is, now we were able to use about *2 times less calls*. In particular, the value $f_1(10001) = 1$ for vector #9.2 was inferred from the fact $f_2(10001) = 1$. However, to restore function f_2 by using f_1, we asked the expert 13 times. That is, we asked him as many times as during the independent restoration of f_2 (i.e., the nested approach was not beneficial in this case). This should not come as a surprise, because the limits described in the previous sections are upper limits. That is, on the average the sequential search is expected to outperform a nonsequential approach, but cases like the last one can still be expected.

The previous analysis shows that, in this particular application, the most effective way was at first to restore f_2 with 13 calls and next to use f_2 values to restore f_1, which required only 7 additional calls. Restoration of f_2 by using information of values of function f_1 required 13 calls, and to restore both functions in E_5 would have required $13 + 13 = 26$ calls. In the sequence of the restoration $\langle f_2, f_1 \rangle$, the total amount of calls to restore both functions is $13 + 7 = 20$ calls, in comparison with $13 + 13 = 26$ calls for independent restoration. We should also add to both cases $12 + 4$ calls which were needed to restore functions φ and ψ. Therefore, in total we needed $20 + 16 = 36$ and $26 + 16 = 42$ calls, respectively. Note that we began from 2×2048 and 2×924 calls for both functions. Our totals of 36 and 42 calls are *about 100 times less than* the number of nonoptimized calls

(i.e., 2×2048) and *about 50 times less than* the upper limit guaranteed according to the Hansel lemma (i.e., the 2×924 calls).

5.5. *A JOINT SEARCH APPROACH FOR NESTED FUNCTIONS*

Next, we study the possibility to decrease the number of calls once more with a joint search approach of f_1 and f_2 by using the following switching strategy:

Step 1. Ask the expert for the value of $f_2(x^{1.1})$ for vector #1.1.

Step 2. If $f_2(x^{1.1}) = 1$, then ask for the first vector of the next chain. That is, ask for the value of $f_2(x^{2.1})$ for vector #2.1.

Otherwise, ask for the value of $f_1(x^{1.2})$.

Step 3. If $f_1(x^{1.1}) = 0$, then ask for the value of $f_1(x^{1.2})$ for vector #1.2.

Otherwise, switch to ask for the value of $f_2(x^{1.1})$.

The generalization of the previous steps for arbitrary $x^{i.k}$ is done with steps A and B. Step A is for f_1 and step B is for f_2. These steps are best described as follows:

Step A. If $f_1(x^{i.k}) = 0$ and vector $\#(i.k)$ is not the last one in the current chain and $f_1(x^{i.k+1}) = e$ (i.e., empty), then ask the expert for the value of $f_1(x^{i.k+1})$. Otherwise, ask for the value for the first vector $x^{i+1.j}$ from the next chain such that $f_1(x^{i+1.j}) = e$.

If $f_1(x^{i.k}) = 1$ and $f_2(x^{i.k}) = e$, then ask for the value of $f_2(x^{i.k})$.

If $f_1(x^{i.k}) = 1$ and $f_2(x^{i.k}) = 0$, then ask for the value of $f_2(y)$, where y is the first vector from the same or the next chain such that $f_2(y) = e$.

Step B. If $f_2(x^{i.k}) = 1$, then ask for the first vector y of the next chain such that $f_2(y) = e$.

If $f_2(x^{i.k}) = 0$ and $f_1(x^{i.k}) = e$, then ask the expert for the first vector y such that $f_1(y) = e$.

The results of applying this strategy to restore the two functions f_1 and f_2 are presented in Table 1 (in columns 10 and 11, respectively). The number of calls to restore both f_1 and f_2 is equal to 22 (see Table 1, columns 10 and 11), by asking for the values of f_1 8 times and for the values of f_2 14 times (see also the values marked with "*" in columns 10 and 11). Note that the previous algorithm required 22 calls.

Next, in Tables 2 and 3, we summarize the various results for interviewing the expert under the different strategies. Table 2 represents numbers of calls for the different kinds of search for the functions f_1, f_2, ψ in E_5 and for φ in E_3. Table 3 represents results for E_{11}. These results summa-

TABLE 2

Comparison of the Results in E_5 and E_3

# (1)	Ways of search (2)	f_1 (3)	f_2 (4)	f_1, f_2 (5)	Index 1 (6)	Index 2 (7)	φ (8)	ψ (9)
1	Nonoptimized search (upper limit)	32	32	64	1	—	8	32
2	Optimal search (upper limit)	20	20	40	1.6	1	6	20
3	Independent search	13	13	26	2.5	1.5	4	12
4	Sequential search $\langle f_1, f_2 \rangle$	13	13	26	2.5	1.5		
5	Sequential search $\langle f_2, f_1 \rangle$	7	13	20	3.2	2.0		
6	Joint search	—	—	22	2.9	1.8		

rize the numbers of calls needed for the full restoration of f_1 and f_2 as functions of 11 binary variables at level 2. In particular, note that the independent search of f_1 required 13 calls in E_5, 12 calls for ψ, and 4 calls for φ, that is, the 29 calls shown in Table 3. The sequential search $\langle f_2, f_1 \rangle$ required the same number of calls for f_2, but only 7 calls for f_1, because we used the same φ and ψ functions found for f_1 (see also Tables 2 and 3).

The total amount shown in column "f_1, f_2" (i.e., under column 5) represents the number of calls to restore both functions. For the last four hierarchical searches in this column, we excluded the nonnecessary calls for the second restoration of ψ and φ, i.e., the 16 calls. For instance, independent search required 29 calls to restore both f_1 and f_2 in E_5, that is, a total of 42 calls in E_{11}, as shown in this column.

Next let us describe the meaning of indexes 1 and 2 in Tables 2 and 3. We denote the upper limit of calls for nonoptimized search as In_1 and the analogous amounts for other ways of the search for both functions (columns 5) as In_i (for $i = 2,3,4,5,6$). In these terms, index $1 = In_1/In_i$ (for $i = 2,3,4,5,6$) and index $2 = In_2/In_i$ (for $i = 3,4,5,6$). In particular, index 1

TABLE 3

Comparison of the Results in E_{11}

# (1)	Ways of search (2)	f_1 (3)	f_2 (4)	f_1, f_2 (5)	Index 1 (6)	Index 2 (7)
1	Nonoptimized search (upper limit)	2,048	2,048	4,096	1	—
2	Optimal search (upper limit)	924	924	1,848	2.22	1
3	Independent search	29	29	42	97.5	48.8
4	Sequential search $\langle f_1, f_2 \rangle$	29	13	42	97.5	48.8
5	Sequential search $\langle f_2, f_1 \rangle$	7	29	36	113.8	51.3
6	Joint search	—	—	38	107.8	48.6

shows that we used 3.2 times less calls than In_1 and 2.0 times less calls than In_2 in E_5, and also 113.8 times less calls than In_1 and 51.3 times less calls than In_2 in E_{11}. These interactive experiments demonstrate the potential for achieving significantly high efficacy of the proposed approach for interactive restoration of monotone Boolean functions.

6. CONCLUDING REMARKS

Some computational experiments (see, for instance, [12, 36]) have shown that it is possible to significantly decrease the number of questions to an oracle in comparison with the full number of questions (which is equal to 2^n) and also in comparison with a guaranteed pessimistic estimation (formula (2.2) in Section 2) for many functions. Some close form results were also obtained for a connected problem of retrieval of maximal upper zero [23]. The results in this paper demonstrate that an interactive approach, based on monotone and hierarchical Boolean functions, has the potential to be very beneficial to interactive machine learning.

The authors are very grateful to Dr. James F. Ruiz, from the Woman's Hospital of Baton Rouge, LA, for his expert assistance in formulating the medical example described in this paper. The first two authors gratefully acknowledge the support from the Office of Naval Research (ONR) Grant N00014-95-1-0639.

REFERENCES

1. D. Angluin, Queries and concept learning, *Mach. Learn.* 2(4):319–342 (1988).
2. N. Bshouty, T. Hancock, L. Hellerstein, and M. Karpinski, An algorithm to learn read-once threshold formulas, and transformations between learning models, *Comput. Complex.* 4:37–6 (1994).
3. V. B. Alekseev, Monotone Boolean functions, in *Encyclopedia of Mathematics*, Kluwer Academic Publishers, Norwell, MA, 1988, vol. 6, pp. 306–307.
4. A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, Learnability and the Vapnik-Chervonenkis dimension, *J. Assoc. Comput. Mach.* 36(4):929–965 (1989).
5. M. Bongard, *Pattern Recognition*, Nauka, Moscow, 1967 (in Russian) [English translation, Spartakos Press, New York, 1970].
6. E. Boros, P. L. Hammer, and T. Ibaraki, Predicting cause-effect relationships from incomplete discrete observations, *SIAM J. Discrete Math.* 7(4):531–543 (1994).
7. D. Cohn, L. Atlas, and R. Ladner, Improving generalizing with active learning, *Mach. Learn.* 15:201–221 (1994).
8. R. Dedekind, Ueber Zerlegungen von Zahlen durch ihre grossten gemeinsamen Teiler, *Fetschrift Hoch. Braunschweig* II:103–148 (1897).
9. T. C. Dietterich and R. S. Michalski, A comparative review of selected methods for learning from examples, in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds., Tioga Publishing Company, Palo Alto, CA, 1983, pp. 41–81.

10. L. M. Fu, Knowledge-based connectionism for revising domain theories, *IEEE Trans. Syst. Man. Cybernet.* 23(1):173–182 (1993).
11. S. Goldman and R. H. Sloan, The power of self-directed learning, *Mach. Learn.* 14:271–294 (1994).
12. Y. Gorbunov and B. Kovalerchuk, An interactive method of monotone Boolean function restoration, *J. Acad. Sci. UzSSR, Eng.* 2:3–6 (1982) (in Russian).
13. P. L. Hammer and E. Boros, Logical analysis: An overview, RUTCOR Res. Rep., Rutgers University, NJ, 1994.
14. G. Hansel, Sur le nombre des fonctions Boolenes monotones den variables, *C. R. Acad. Sci. Paris* 262(20):1088–1090 (1966).
15. D. Haussler, Quantifying inductive bias: AI learning algorithms and Valiant's learning framework, *Artif. Intell.* 36:177–221 (1988).
16. D. Haussler and M. Warmuth, The probably approximately correct (PAC) and other learning models, in *Foundation of Knowledge Acquisition: Machine Learning*, A. L. Meyrowitz and S. Chipman, Eds., Kluwer Academic Publishers, Norwell, MA, 1993, pp. 291–312.
17. K. Hattoru and Y. Torri, Effective algorithms for the nearest neighbor method in the clustering problem, *Patt. Recog.* 26(5):741–746 (1993).
18. A. P. Kamath, N. K. Karmakar, K. G. Ramakrishnan, and M. G. C. Resende, A continuous approach to inductive inference, *Math. Progr.* 57:215–238 (1992).
19. A. P. Kamath, N. K. Karmakar, K. G. Ramakrishnan, and M. G. C. Resende, An interior point approach to Boolean vector synthesis, in *Proceedings of the 36-th MSCAS*, 1994, pp. 1–5.
20. B. Kamgar-Parsi and L. N. Kanal, An improved branch-and-bound algorithm for computing k-nearest neighbors, *Patt. Recog. Lett.* 3:7–12 (1985).
21. D. Kleitman, On Dedekind's problem: The number of monotone Boolean functions, *Proc. Am. Math. Soc.* 21:677–682 (1969).
22. V. K. Korobkov, On monotone Boolean functions of algebra logic, in *Problemy Cybernetiki*, Nauka, Moscow, vol. 13, pp. 5–28, 1965 (in Russian).
23. B. Kovalerchuk and V. Lavkov, Retrieval of the maximum upper zero for minimizing the number of attributes in regression analysis, *USSR Computat. Math. Math. Phys.* 24(4):170–175 (1984).
24. B. Kovalerchuk, E. Triantaphyllou, and E. Vityaev, Monotone Boolean functions learning techniques integrated with user interaction, in *Proceedings of Workshop "Learning from Examples vs. Programming by Demonstration," 12th International Conference on Machine Learning*, Tahoe City, CA, 1995, pp. 41–48.
25. B. Kovalerchuk, E. Triantaphyllou, and J. F. Ruiz, Monotonicity and logical analysis of data: A mechanism of evaluation of mammographic and clinical data, in *SCAR '96, Computer Applications to Assist Radiology*, Symposia Foundation, Carlsbad, CA, 1996, to appear.
26. T. Kurita, An efficient agglomerative clustering algorithm using a heap, *Patt. Recog.* 24(3):205–209 (1991).
27. O. L. Mangasarian, W. N. Street, and W. H. Woldberg, Breast cancer diagnosis and prognosis via linear programming, *Oper. Res.* 43(4):570–577 (1995).
28. T. Mitchell, The need for biases in learning generalizations, Tech. Rep. CBM-TR-117, Rutgers University, New Brunswick, NJ, 1980.
29. P. M. Murphy and D. W. Aha, UCI repository of machine learning databases: Machine-readable data repository, Dept. of Inform. Comput. Sci., University of California, Irvine, CA, 1994.
30. B. K. Natarajan, On learning sets and functions, *Mach. Learn.* 4(1):123–133 (1989).

31. S. Rudeanu, *Boolean Functions and Equations*, North-Holland, Amsterdam, 1974.
32. R. Schapire, *Design and Analysis of Efficient Learning Algorithms*, MIT Press, Cambridge, MA, 1992.
33. J. W. Shavlik, Combining symbolic and neural learning, *Mach. Learn.* 14:321–331 (1994).
34. E. Triantaphyllou, A. L. Soyster, and S. R. T. Kumara, Generating logical expressions from positive and negative examples via a branch-and-bound approach, *Comput. Oper. Res.* 21(2):185–197 (1994).
35. E. Triantaphyllou, Inference of a minimum size Boolean function from examples by using a new efficient branch-and-bound approach, *J. Global Optim.* 5(1):69–94 (1994).
36. E. Triantaphyllou and A. Soyster, An approach to guided learning of Boolean functions, *Math. Comput. Model.* 23(3):69–86 (1995).
37. V. N. Vapnik, *Estimating of Dependencies Based on Empirical Data*, Springer-Verlag, New York, 1982.
38. L. G. Valiant, A theory of learnable, *Commun. ACM* 27:1134–1142 (1984).
39. E. Vityaev and A. Moskvitin, Introduction to discovery theory. Program system: DISCOVERY, Logical methods in informatics, in *Computational Systems* (*Vychislitel'nye sistemy*), Institute of Mathematics, Russian Academy of Science, Novosibirsk, 1993, no. 148, pp. 117–163 (in Russian).
40. W. W. Woldberg and O. L. Mangasarian, Multisurface method of pattern separation for medical diagnosis applied to breast cytology, *Proc. Nat. Acad. Sci. USA*, 87(23):9193–9196 (1990).
41. S. Yablonskii, *Introduction to Discrete Mathematics*, Nauka, Moscow, 1986 (in Russian).
42. N. Zagoruiko, *Empirical Forecast*, Nauka, Novosibirsk, 1979 (in Russian).

Received 1 September 1995