**Published in:**

*Mathematical and Computer Modelling,* **Vol. 23, No. 3, pp. 69-86, 1996.**

# An Approach to Guided Learning of Boolean Functions

*Last revision: April 1995*

E. TRIANTAPHYLLOU
Department of Industrial and Manufacturing Systems Engineering
3128 CEBA Building
Louisiana State University
Baton Rouge, LA 70803-6409, U.S.A.
E-mail: IETRIAN@LSUVM.SNCC.LSU.EDU
Web: http://www.imse.lsu.edu/vangelis

A. L. SOYSTER
Department of Industrial and Management Systems Engineering
207 Hammond Building
Penn State University
University Park, PA 16802, U.S.A.
E-mail: ALSIE@ENGR.PSU.EDU

**Abstract--** A critical aspect in the problem of inductive inference is the number of examples needed to accurately infer a Boolean function from positive and negative examples. In this paper we develop an approach for deriving a sequence of examples for this problem. Some computer experiments indicate that, on the average, examples derived according to the proposed approach lead to the inference of the correct function considerably faster than when examples are derived in a random order.

**Key words:** Inductive inference, Boolean functions, learning from examples, artificial intelligence, clause satisfiability problem.

## 1. INTRODUCTION

One of the most critical problems in the development of rule-based systems is that of learning. Learning is the main property which characterizes intelligent behavior in a system. This is due to the belief that any intelligent system should be able to improve its performance over time. However, very often knowledge about real life situations is erroneous and incomplete. In such cases a learning mechanism can utilize feed-back and try to improve the performance of the system.

For these reasons, learning from examples has attracted the interest of many researchers in recent years. In the typical learning problem of this type, both positive and negative examples are available and the main goal is to determine a Boolean expression (that is, a set of logical rules or clauses) which accepts all the positive examples, while it rejects all the negative examples. This kind of learning has been examined intensively in the last years (see, for instance, [1], [2], [3], [4], [5], [6], and [7]). Typically the rule base of a system is expressed as a Boolean function either in the conjunctive normal form (CNF) or in the disjunctive normal form (DNF) (see, for instance, [8], [9], [10], [11], [12], [13], [14], [3], [15], and [16]).

A considerable amount of related research is today known as the **PAC (Probably Approximately Correct)** learning theory (see, for instance, [7], [17], and [18]). The central idea of the PAC model is that successful learning of an unknown target concept should entail obtaining, with high probability, a hypothesis that is a good approximation of the target concept (hence the term: *probably approximately correct*). The error associated with the approximation of the target concept is defined as the probability that the proposed concept (denoted as $h$) and the target concept (denoted as $c$) will disagree on classifying a new example drawn randomly from unclassified examples. Later in this paper this notion of error is used frequently and is related to another concept used extensively in this paper called **accuracy** rate. The hypothesis $h$ is a good approximation of the target concept if the previous error is small (less than some quantity $\epsilon$, where: $1 > \epsilon > 0$).

In the same framework of thought, a learning algorithm is then a computational procedure which takes a sample of random positive and negative examples of the target concept $c$ and returns a hypothesis $h$. In the literature a learning algorithm $A$ is a PAC algorithm if for all positive numbers $\epsilon$ and $\delta$ (where: $1 > \epsilon, \delta > 0$), when $A$ runs and accesses unclassified examples then it eventually halts and outputs a concept $h$ with probability at least $1 - \delta$ and error at most $\epsilon$ [19].

Conjunctive concepts are properly PAC learnable [7]. However, the class of concepts in the form of the disjunction of two conjunctions is not properly PAC learnable [5]. The same is also true for the class of existential conjunctive concepts on structural instance spaces with two objects [20]. The classes of $k$-DNF, $k$-CNF, and $k$-decision lists are properly PAC learnable for each fixed $k$ [15], [21], but it is **unknown** whether the classes of all DNF, or CNF functions are PAC learnable [18]. Note that the present paper deals with Boolean functions in CNF or DNF form. Therefore, the learning algorithms used later in this paper in conjunction with the proposed guided learning approach are not known whether they are PAC learnable. Also, Mansour in [22] gives an $n^{O(loglogn)}$ algorithm for learning DNF formulas (however, **not** of minimal size) under a uniform distribution using membership queries.

Besides improving the performance of an existing system, a learning mechanism can assist in creating the initial rule base. That is, it can assist in the knowledge acquisition phase. By asking the human expert a short sequence of key questions, we hope that the rules of a rule-based system can be configured accurately and efficiently.

In the context of this paper these questions refer to the **classification of examples**. That is, the human expert is presented with a new example, one at a time. Then, the (also called *"oracle"* in the literature) expert is asked to classify this example either as positive or as negative. Although the rules may not be explicitly known to the expert, it is assumed that the expert can classify new examples correctly (an example is positive if it satisfies all the rules, otherwise it is negative). The **inductive inference problem** is to derive the *"hidden"* rules (also called the *"hidden logic"* system) from the classifications of sampled examples.

***The problem examined in this paper is how to derive new examples.*** It is assumed that two initial sets of positive and negative examples are given. Since these initial sets are a small sample of all the possibilities, new examples may be required. An effective sequence of new examples should be able to lead to the accurate inference of the *"hidden logic"* system by considering relatively few new examples.

This paper is organized as follows. The following section describes some definitions and terminology. The third section sets forth a rigorous problem definition. Sections 4, 5, and 6 illustrate the proposed methodology for deriving new examples. Section 7 presents some computational results. Finally, the last section summarizes the main points and conclusions of this paper.

## 2. SOME DEFINITIONS AND TERMINOLOGY

In this section we introduce some terminology. Let $\{A_1, A_2, A_3, \ldots, A_t\}$ be a set of t Boolean **predicates** or **atoms**. That is, each $A_i$ $(i=1,2,3,\ldots,t)$ can be either true (denoted by 1) or false (denoted by 0). Let $F$ be a **Boolean function** over $\{A_1, A_2, A_3, \ldots, A_t\}$. That is, $F$ is a mapping from $\{0,1\}^t \rightarrow \{0,1\}$ which determines for each combination of truth values of the arguments $A_1, A_2, A_3, \ldots, A_t$ of $F$, whether $F$ is true or false (denoted as 1 and 0, respectively).

For each such Boolean function $F$, the **positive examples** are the vectors $v \in \{0,1\}^t$ such that $F(v) = 1$.

Similarly, the **negative examples** are the vectors $v \in \{0,1\}^t$ such that $F(v) = 0$. Hence, a vector $v \in \{0,1\}^t$ is either a positive or a negative example. Equivalently, we say that a vector $v \in \{0,1\}^t$ is **accepted** (or **rejected**) by a Boolean function if and only if the vector *v* is a positive (or a negative) example of *F*, respectively. In the present paper, a set of positive examples will be denoted as $E^+$. Similarly, a set of negative examples will be denoted as $E^-$.

In the previous paragraphs, a Boolean function *F* is assumed to be of any form. However, in propositional calculus it is convenient to represent Boolean functions using the **conjunctive normal form (CNF)** or the **disjunctive normal form (DNF)**. Peysakh in [23] describes an algorithm for converting any Boolean expression into CNF.

A Boolean expression is in CNF form if (where $a_j$ is either $A_j$ or $\bar{A}_j$):

$$\bigwedge_{i=1}^{N} \left( \bigvee_{j=1}^{M_i} a_j \right)$$

Similarly, a Boolean expression is in DNF if it is in the form:

$$\bigvee_{i=1}^{N} \left( \bigwedge_{j=1}^{M_i} a_j \right)$$

Since the rule base of any expert system contains a finite selection of rules and any rule can be reduced to clausal form [11], it follows that any propositional rule base can be expressed as a finite set of disjunctions (clauses in CNF form). Therefore, we can assume that any such system is in CNF form.

## 3. Problem Description.

At this point suppose that there exists a *"hidden logic"* system. That is, there is a system which we would like to infer from collections of positive and negative examples. Although we **cannot explicitly** identify the rules of the *"hidden system"*, it is assumed that it is possible to correctly classify any new examples according to these rules. This can occur, for instance, by interviewing a human expert.

To help fix ideas, suppose that the following represents the *"hidden logic"* system:

$$(\bar{A}_1 \vee \bar{A}_4 \vee A_6) \wedge (\bar{A}_2 \vee A_8) \wedge (A_2) .$$

This system is considered to be **unknown** to the user. By *user* we mean here the person (or another computer system) which wants to infer the *"hidden logic"* system from collections of positive and negative examples. Next, let the following sets $E^+$ and $E^-$ represent two collections of positive and negative examples, respectively. These are the examples which an expert has already classified.

$$E^+ = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \qquad E^- = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Given the above examples, we want to determine a set of clauses (i.e., a Boolean function) which correctly classify the previous examples.

An early definition of the clause inference from examples problem was given Bongard in [24] (the original book was published in Russian in 1967). Some recent developments can be found in [25], and [3]. In [25] an algorithm which infers CNF systems from positive and negative examples is developed. In that approach, CNF clauses are generated in a way which attempts to **minimize** the number of CNF clauses that constitute the recommended CNF system. The strategy followed there is called the **One Clause At a Time (or OCAT)** approach and was combined with a branch-and-bound algorithm. A new and more efficient branch-and-bound algorithm is described in [26].

Another related algorithm is presented in [3]. This algorithm formulates the clause inference problem as a **clause satisfiability** problem. In turn, this satisfiability problem is solved by using an interior point method proposed by Karmakar and his associates in [27]. Furthermore, by applying the second algorithm iteratively it is possible to determine a DNF system of minimum size (i.e., of minimum number of conjunctions).

When the OCAT approach is applied on the previous $E^+$ and $E^-$ sets of examples, the following CNF system is derived (we call it system $S_{SAMPLE}$ to emphasize that the rules have been derived from sampled data):

$$(\bar{A}_3 \vee A_8) \wedge (A_2)\,. \tag{1.1}$$

In [28] it is demonstrated that any algorithm which infers a CNF (or DNF) system from examples can also infer a DNF (or CNF) system by performing some simple transformations on the original data. Furthermore, some ways for solving large scale clause inference problems are discussed in [29]. The proposed system (1.1) may or may not be a good approximation of the *"hidden logic"* system. Recall that any sampling process is subject to random variation.

Suppose that the user can supply the expert with additional examples for correct classification. Then, the **main problem** examined in this paper is **how to generate the next example.** One obvious approach is to generate the next example **randomly**. However, this may result in generating many examples and still not achieving a good approximation of the unknown system. It is obviously desirable to consider a sequence of new examples which can lead to a good approximation of the unknown system as quickly as possible.

When a new example is considered, it is given to the expert for the correct classification. Two situations can occur. First, the current Boolean function (which is attempting to represent the unknown *"hidden logic"*) classifies the new example in a manner **identical** with the expert (who always correctly classifies each example). In the second case, the new example is classified in the **opposite** way by the expert and the current Boolean function. If the current Boolean function is not yet a good approximation of the *"hidden logic"*, then the last case is the most desirable scenario. This is true, because in this case one can re-execute a clause inference algorithm (for instance, the OCAT approach) again and, hopefully, derive a closer approximation of the unknown system.

If the current version of the Boolean function is an inaccurate approximation of the *"hidden logic"* and one generates new examples which fail to reveal any contradictions, then additional costs are incurred in classifying new examples, but **no improvement** is gained. Clearly, it is desirable to use a strategy for determining the next example, such that any possible contradiction between the current version of the Boolean function and the *"hidden logic"* will **surface early in the interviewing process.** The next section presents the development of such a strategy.

## 4. THE PROPOSED APPROACH

Consider two sets of positive and negative examples, $E^+$ and $E^-$, defined on $t$ atoms. Let $S_{SAMPLE}$ denote a rule base (i.e., a Boolean function) that correctly classifies the sample data, i.e. the examples in $E^+$ are classified as positive and the examples in $E^-$ are classified as negative (one such Boolean function can be obtained via the methods described in [3], [25], [26], and recently in [30]). When the proposed guided learning strategy is applied, then the derived system will be denoted as $S_{GUIDED}$. Also, define $S_{HIDDEN}$ as the *"hidden logic"* Boolean function and $\overline{S}_{HIDDEN}$ as the complement of $S_{HIDDEN}$. Hence, if $S_{HIDDEN}$ is

$$(A_1 \vee A_2) \wedge (\overline{A}_2 \vee A_3),$$

then $\overline{S}_{HIDDEN}$ is

$$\overline{(A_1 \vee A_2)} \vee \overline{(\overline{A}_2 \vee A_3)}.$$

Our objective is to sequentially modify and improve $S_{GUIDED}$ so that

$$S_{GUIDED} \dashrightarrow S_{HIDDEN},$$

where additional examples are generated and included either in $E^+$ or in $E^-$. If the sequence of distinct examples generated so far is denoted as $v_1, v_2, v_3, \ldots, v_k$, then at least when $k = 2^t$, one must, by definition (since all possible examples have been generated) obtain

$$S_{GUIDED} = S_{HIDDEN}.$$

The objective of our algorithm is to choose a sequence of examples so that

$$S_{GUIDED} \approx S_{HIDDEN},$$

even when $k$ is rather small (maybe only a tiny fraction of $2^t$). We view the problem from a local perspective only. ***In particular, if k examples have already been generated, then what should be the $k+1$st example?***

The method by which we propose to select the $k+1$st example is based on the observation that for any example $v$, either $S_{HIDDEN}$ or $\overline{S}_{HIDDEN}$ must classify the example as positive, but **not** both. Denote $S_{HIDDEN}(v) = 1$ if Boolean function $S_{HIDDEN}$ classifies the example $v$ as positive, and $S_{HIDDEN}(v) = 0$ if it classifies it as negative. Then, for any example $v$ the following relation is always true

$$S_{HIDDEN}(v) + \overline{S}_{HIDDEN}(v) = 1.$$

Next, consider a Boolean function, $S_{GUIDED}$, determined by sampling $k$ examples and generating a set of clauses which correctly classifies the $k$ examples. $S_{GUIDED}$ is an approximation to $S_{HIDDEN}$. The OCAT (*One Clause At a Time*) procedure described in [25] and in [26] is one strategy for determining $S_{GUIDED}$. However, an algorithm such as OCAT could also be applied to a type of **dual problem**, i.e. a problem in which the positive examples are treated as negative and vice-versa. Let $S_{\text{R-GUIDED}}$ be a Boolean function generated when the $k$ examples are assigned **reverse** truth values. That is, the positive examples are treated as negative and the negative examples as positive. In this manner, $S_{\text{R-GUIDED}}$ would be an approximation to $\overline{S}_{\text{HIDDEN}}$. If, indeed $k = 2^t$, then

$$S_{\text{GUIDED}}(v) + S_{\text{R-GUIDED}}(v) = 1, \tag{4.1}$$

for all examples $v$. However, in general for $k < 2^t$ one should expect that examples will exist for which the sum in (4.1) will be 0 or 2, i.e. some example $v$ will be classified either as negative (sum is equal to 0) or as positive (sum is equal to 2) by **both** Boolean functions. Such an example is the key to our approach. The existence of such an example means that exactly one of our two example generated Boolean functions (i.e., $S_{\text{GUIDED}}$ and $S_{\text{R-GUIDED}}$) is in error. Either $S_{\text{GUIDED}}$ or $S_{\text{R-GUIDED}}$ must be modified to **correctly** classify the new example. These observations are summarized in the following theorem:

***THEOREM 1:***

*Suppose that there exists an example $v \in \{0,1\}^t$ such that:*

$$S_{GUIDED}(v) + S_{R\text{-}GUIDED}(v) = 0 \quad \text{or:} \tag{4.2a}$$

$$S_{GUIDED}(v) + S_{R\text{-}GUIDED}(v) = 2. \tag{4.2b}$$

*Furthermore, suppose that the example $v$ is classified by the expert as either positive or negative. Then, one and only one of the following situations is true:*

*a) If (4.2a) holds and $v$ is a positive example, then system $S_{GUIDED}$ is not valid.*

*b) If (4.2a) holds and $v$ is a negative example, then system $S_{R\text{-}GUIDED}$ is not valid.*

*c) If (4.2b) holds and $v$ is a positive example, then system $S_{R\text{-}GUIDED}$ is not valid.*

*d) If (4.2b) holds and $v$ is a negative example, then system $S_{GUIDED}$ is not valid.*

Therefore, the overall strategy, starting with two Boolean functions, is to attempt to generate a sequence of new examples $v_{k+1}, v_{k+2}, v_{k+3}, \ldots, v_m$, where each example is appropriately classified, as positive or negative, by the expert. Each additional example should have the property that it invalidates either $S_{\text{GUIDED}}$ or $S_{\text{R-GUIDED}}$, i.e. one of the two Boolean functions must be modified. In doing so, it is expected that $S_{\text{GUIDED}}$ and $S_{\text{R-GUIDED}}$

become more closely aligned with $S_{HIDDEN}$ and $\overline{S}_{HIDDEN}$, respectively.

How does one find an example that invalidates either $S_{GUIDED}$ or $S_{R\text{-}GUIDED}$? Conceptually it is quite simple. One strategy is to formulate and solve at most two **clause satisfiability problems**. The satisfiability (or SAT) problem is NP-complete and can be defined as follows (see, for instance, [10] and [14]):

*Consider the m CNF clauses $C_1, C_2, C_3, \ldots, C_m$ involving the t atoms $A_1, A_2, A_3, \ldots, A_t$, and the Boolean expression:* $C_1 \wedge C_2 \wedge C_3 \wedge \ldots \wedge C_m$, *The expression is **satisfiable** if there exists an assignment of truth values which makes the Boolean expression true.*

The clause satisfiability problem has been examined with considerable success [10], and [11]. A recent development reported in [3] uses an interior point algorithm developed by Karmakar and his associates in [27] with considerable success. Also, some problem preprocessing techniques can be found in [9].

The two satisfiability problems of interest in this paper are as follows: Determine an example $\overline{v}$ which results in a truth value TRUE for

$$\overline{S}_{SAMPLE}(\overline{v}) \wedge \overline{S}_{R-SAMPLE}(\overline{v}), \tag{4.3}$$

or:

$$S_{SAMPLE}(\overline{v}) \wedge S_{R-SAMPLE}(\overline{v}) \tag{4.4}$$

If (4.3) is TRUE (i.e., satisfied), then $\overline{v}$ is evaluated as negative by both systems (Boolean functions), and if (4.4) is TRUE, then $\overline{v}$ is evaluated as positive by both systems. Observe that relations (4.3) and (4.4) are **equivalent** to relations (4.2a) and (4.2b), respectively.

If an example is found which satisfies either (4.3) or (4.4), then one of the two Boolean functions is modified and the same process is repeated. To illustrate, suppose $\overline{v}$ satisfies (4.4) and the expert classifies the example $\overline{v}$ as positive (TRUE). In this case $S_{R\text{-}GUIDED}$ does not classify $\overline{v}$ correctly and thus it must be updated. If the expert classifies $\overline{v}$ as negative (FALSE), then the example invalidates $S_{GUIDED}$ and thus $S_{GUIDED}$ must be updated. If the example $\overline{v}$ satisfies (4.3), a similar analysis prevails. (Note that if we find an example $\overline{v}$ which satisfies (4.3), then there is no need to also search for an example which satisfies (4.4).) If no example can be found to satisfy (4.3), then we search to find an example which satisfies (4.4).

But, suppose that there are no examples which satisfy (4.3) or (4.4). Does this mean that

$$S_{\text{GUIDED}} \equiv S_{\text{HIDDEN}} \ ?$$

Unfortunately, **<u>the answer is no</u>**. As a trivial demonstration of why this is true, consider the case in which $S_{\text{HIDDEN}} = (A_1 \vee A_2)$. Next, consider the input examples to be defined as follows: $E^+ = [1\ 0]$ and $E^- = [0\ 0]$. Then, OCAT returns $S_{\text{GUIDED}} = A_1$, and $S_{\text{R-GUIDED}} = \overline{A}_1$ . Clearly, although neither relation (4.3) nor (4.4) is satisfied in this case $S_{\text{GUIDED}}$ is different than $S_{\text{HIDDEN}}$. In cases like this we revert to a **<u>random</u>** search process. Examples are randomly generated, say $v_{k+1}, v_{k+2}, v_{k+3}, \ldots, v_m$. The expert appropriately classifies $v_{k+1}$ (as positive or as negative) and one evaluates

$$S_{\text{GUIDED}}(v_{k+1}) \tag{4.5}$$

and

$$S_{\text{R-GUIDED}}(v_{k+1}), \tag{4.6}$$

for consistency. That is, if $v_{k+1}$ is positive then (4.5) should be true and (4.6) false, and if $v_{k+1}$ is negative then the converse must be true. This raises the question of a termination criterion. How long should one generate new examples?

Clearly, there are two factors that one needs to take under consideration. One is the **<u>cost</u>** of generating and classifying new examples. The second factor is the **<u>desired accuracy</u>**. In general, one expects that the more examples one uses to infer a set of clauses, the more accurate the inferred system should be. Therefore, it is recommended that if no inconsistency is determined for some large value of $m$ (which value depends on the cost of classifying new examples), the process is terminated and $S_{\text{GUIDED}}$ is our approximation to the *"hidden logic"*. The proposed strategy is depicted in Figure 1, and will also be illustrated in the example described in section 6.

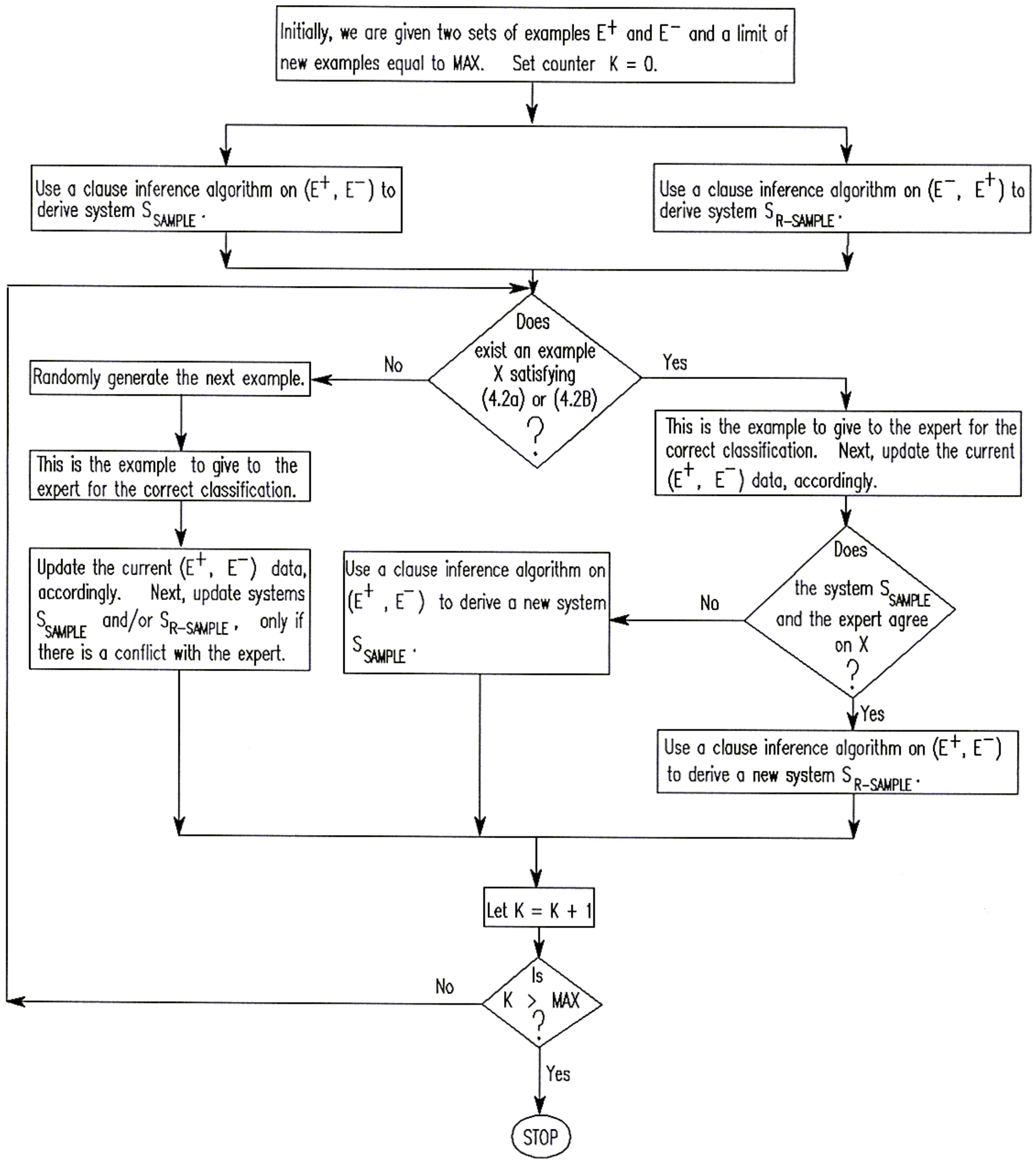
Initially, we are given two sets of examples $E^+$ and $E^-$ and a limit of new examples equal to MAX. Set counter K = 0.
Use a clause inference algorithm on $(E^+, E^-)$ to derive system $S_{SAMPLE}$.
Use a clause inference algorithm on $(E^-, E^+)$ to derive system $S_{R-SAMPLE}$.
Does exist an example X satisfying (4.2a) or (4.2B) ?
No
Yes
Randomly generate the next example.
This is the example to give to the expert for the correct classification.
This is the example to give to the expert for the correct classification. Next, update the current $(E^+, E^-)$ data, accordingly.
Update the current $(E^+, E^-)$ data, accordingly. Next, update systems $S_{SAMPLE}$ and/or $S_{R-SAMPLE}$, only if there is a conflict with the expert.
Use a clause inference algorithm on $(E^+, E^-)$ to derive a new system $S_{SAMPLE}$.
Does the system $S_{SAMPLE}$ and the expert agree on X ?
No
Yes
Use a clause inference algorithm on $(E^+, E^-)$ to derive a new system $S_{R-SAMPLE}$.
Let K = K + 1
Is K > MAX ?
No
Yes
STOP

**Figure 1.** Flowchart of the Proposed Strategy for Guided Learning.

A confidence interval on the probability that the derived system $S_{GUIDED}$ will disagree with the *"hidden logic"* can be found as follows: Suppose that $S_{GUIDED}$ has just been redefined when the *i-th* example was presented (that is, $i = *E^+* + *E^-*$). Let the number of disagreements between $S_{GUIDED}$ and $S_{HIDDEN}$ be $r$. Apparently, these disagreements are among the remaining $2^t - i$ (where $t$ is the number of atoms) unclassified examples. A reasonable model of total ignorance of the outcomes (i.e., the way the remaining examples are classified) of the remaining steps is that all permutations of these outcomes are equally likely. The probability, denoted as $Q_r(x)$, of $x$ or more additional steps without a disagreement is then:

$$\frac{\binom{2^t-i-r}{x}}{\binom{2^t-i}{x}} = \frac{(2^t-i-x)\ldots(2^t-i-x-r+1)}{(2^t-i)\ldots(2^t-i-r+1)} = \frac{(2^t-i-r)\ldots(2^t-i-r-x+1)}{(2^t-i)\ldots(2^t-i-x+1)}.$$

Note that this probability, besides of the values of $x$ and $r$, also depends on $i$.

Given $x$, then find the greatest $r$ such that $Q_r(x) \geq \alpha$ (where $\alpha$ might be 0.05 to correspond to a 5% level of significance or a 95% level of confidence). This is (under the random permutation model) an upper confidence bound on $r$. Observe that the previous model did not consider any pattern of agreements in the $i$ examples already classified by the *"hidden logic"*. Therefore, it is possible to derive tighter bounds, when all issues are considered.

## 5. NUMBER OF CANDIDATE SOLUTIONS

An important issue related to this problem is to determine the number of all possible **distinct** systems which satisfy the requirements of two sets of examples $E^+$ and $E^-$. Two systems, defined on $t$ atoms, are called distinct if they are not equivalent. That is, if they classify the examples in $\{0,1\}^t$ differently. Suppose that $*E^+* + *E^-* < 2^t$. Then, ***the number of distinct systems which satisfy the requirements of the current positive and negative examples is equal to the number of all possible ways that the remaining examples can be divided into positive and negative examples***.

Let L denote the number of the remaining examples. That is, $L = 2^t - (*E^+* + *E^-*)$. Since each row of the truth table for the L unclassified examples can be independently filled with 0 or 1, the answer to the previous question is $2^L$. Therefore, the following lemma 1 is true:

***LEMMA 1:***

*Suppose that $E^+$ and $E^-$ are the sets with the positive and negative examples, respectively. Then $K$, the number of distinct systems which satisfy the requirements of these examples, is given by the following formula:*

$$K = 2^L, \quad \text{where } L = 2^t - (*E^+* + *E^-*).$$

The above number $K$ is extremely large even for small values of $t$. This value $K$ is the size of the hypothesis space and is used in the next section to quantify the **space complexity** of a learning algorithm. In the next section these ideas are further illustrated via an example.

## 6. AN ILLUSTRATIVE EXAMPLE

Consider the two collections of positive and negative examples which were given in the third section. Recall that it was assumed that the system $S_{\text{HIDDEN}}$ (i.e., the *"hidden logic"* system) is as follows:

$$(\bar{A}_1 \vee \bar{A}_4 \vee A_6) \wedge (\bar{A}_2 \vee A_8) \wedge (A_2).$$

In this illustrative example, we use the OCAT approach ([25], [26]) in order to derive a CNF system from the given examples. When the OCAT approach is applied on the $(E^+, E^-)$ sets of examples, the following system (Boolean function in CNF), $S_{\text{GUIDED}}$, is derived:

$$(\bar{A}_3 \vee A_8) \wedge (A_2).$$

From lemma 1 it follows that the total number of systems which satisfy the requirements of these examples (and hence, are candidates to be the *"hidden logic"*) is $2^{246}$. This is an astronomically large number. The value 246 in the previous expression is derived from: $246 = 2^8 - (3 + 7)$. Next, consider the system which is derived when the positive examples are treated as negative and the negative examples as positive. When the OCAT approach is applied on the $(E^-, E^+)$ data, then the following system (Boolean function in CNF), $S_{\text{R-GUIDED}}$, is derived:

$$(\bar{A}_2 \vee \bar{A}_8).$$

The next issue to investigate is to see whether there is an example which satisfies relation (4.2a) or (4.2b). Observe that the new example (i.e., still unclassified) (0 1 0 1 1 0 1 0) is classified as **positive by both systems**. That is, this example makes (4.2b) to be true. This example can be determined by finding a feasible solution of the clause satisfiability problem formed when the two systems $S_{GUIDED}$ and $S_{R\text{-}GUIDED}$ are taken together. That is, the satisfiability problem is:

$$S_{SAMPLE} \wedge S_{R-SAMPLE}$$
$$or:$$
$$((\bar{A}_3 \vee A_8) \wedge (A_2)) \wedge (\bar{A}_2 \vee \bar{A}_8) =$$
$$(\bar{A}_3 \vee A_8) \wedge (A_2) \wedge (\bar{A}_2 \vee \bar{A}_8).$$

Following theorem 1, either the system $S_{GUIDED}$ or the system $S_{R\text{-}GUIDED}$ will be revised and updated. Now suppose that the expert classifies the new example as a **negative** example. Hence, it follows that system $S_{GUIDED}$ is invalid.

Next, this new example is added to the current collection of the negative examples and the OCAT approach is reapplied on the updated input data. The **new (i.e., updated) version** of the system $S_{GUIDED}$ is as follows:

$$(A_8) \wedge (A_2).$$

Since the new example did not reveal any inaccuracies for system $S_{R\text{-}GUIDED}$, this system needs no modification at this time. Observe that the new version of the system $S_{GUIDED}$ and the current version of the system $S_{R\text{-}GUIDED}$ classify all example in $\{0,1\}^8$ in exactly the opposite manner (i.e., they are the complement of each other). Therefore, no new examples can be determined as above. As indicated earlier, this does not necessarily imply that the current version of the system $S_{GUIDED}$ is equivalent to the *"hidden logic"* system, $S_{HIDDEN}$. In situations like the above, it is proposed that the next example be generated **randomly.** Figure 1 summarizes the main steps of the proposed guided learning strategy.

At this point it is interesting to make some additional observations. In this illustrative example the structure of the *"hidden logic"* system is known. Therefore, it is possible to estimate how closely the proposed system (i.e., system $S_{GUIDED}$) approximates the *"hidden logic"* system $S_{HIDDEN}$. To accomplish this task in this illustrative example, all the remaining 246 ( $= 2^8 - (3+7)$) unclassified examples have been evaluated by $S_{GUIDED}$ and $S_{HIDDEN}$ and compared. The 246 examples, as evaluated by $S_{GUIDED}$ and $S_{HIDDEN}$, agree 84% of the time. Hence, if a new example is chosen at random, there is 84% chance that $S_{GUIDED}$ will correctly classify it. Furthermore, when the **updated version** of the system $S_{GUIDED}$ is compared with the *"hidden logic"* system $S_{HIDDEN}$ they agree 97% of the time.

It should also be stated here that when a new example is considered and system $S_{GUIDED}$ is updated, the new system is **<u>not always</u>** a closer approximation of the *"hidden logic"*. It is sometimes possible that the updated system to be a **<u>worse</u>** approximation of the *"hidden logic"*. Hence, convergence in terms of this measure of performance is not necessarily uniform.

The size of the hypothesis space is influential in determining the **<u>sample complexity</u>** of a learning algorithm. That is, the number of examples needed to accurately approximate a target concept. The presence of **<u>bias</u>** in the selection of a hypothesis from the hypothesis space can be beneficial in reducing the sample complexity of a learning algorithm [31], [32]. Usually the amount of bias in the hypothesis space *H* is measured in terms of the *Vapnik-Chervonenkis dimension*, denoted as *VCdim(H)* [33], [34]. A well known theoretical result regarding the *VCdim(H)* is due to [33] and states that the sample complexity is at most:

$$\frac{1}{\varepsilon( - \sqrt{\varepsilon})}\left(2\,VCdim(H)\,\mathbf{ln}\frac{6}{\varepsilon} + \mathbf{ln}\frac{2}{\delta}\right).$$

This is better than some other bounds given in [36]. However, the previous bound is still an overestimate [18].

The proposed strategy makes no assumption regarding the hypothesis space. In the computational experiments reported in this paper, the OCAT approach was used to infer a Boolean function from positive and negative examples. The OCAT approach has a tendency to return CNF (or DNF) functions with very few terms (i.e., disjunctions or conjunctions, respectively). Therefore, when the OCAT approach is combined with the proposed guided learning strategy, only then the hypothesis space is biased in favor of functions with small representations. That is, the proposed guided learning strategy makes no assumption regarding the hypothesis space. However, the behavior of the proposed strategy can be influenced by the nature of the algorithm used to infer the Boolean function. Why inferring functions with a few terms is desirable, is best explained in [25] and [3].

The OCAT and SAT approaches are NP-complete (for more details see [25] and [3], respectively). The present paper **does not** deal with the problem of inferring a Boolean function from collections of examples. Instead, it examines the problem of what should be next example to be considered if one wishes to correctly infer a *"hidden logic"* by using a short sequence of new examples.

In this paper we do not limit the form of the target Boolean function. The OCAT and SAT approaches used to illustrate the function of the proposed guided learning strategy, do not restrict themselves in deriving *k*-CNF or *k*-DNF functions. If the restriction to derive a *k*-CNF or *k*-DNF function is imposed, then the developments are exactly the same as with the unrestricted case.

The present guided learning problem mentions the issue whether one can infer a Boolean function when a new example is considered. Obviously, a brutal force way is to solve the function inference problem from the beginning. A more efficient way is to try to devise an **incremental** approach in inferring a function when only one new example is introduced and the OCAT or the SAT approaches are used. For instance, this is the case in [37] where an incremental approach to the original *ID3* algorithm [6] is described. Although this is an interesting issue, it is out of the scope of this paper.

## 7. SOME COMPUTATIONAL RESULTS

A number of computer experiments was conducted in order to investigate the effectiveness of the proposed strategy compared with random input learning (that is, when new examples are generated randomly). These experiments were approached in a manner similar to the illustrative example of the previous section. At first, a *"hidden logic"* was generated. The *"hidden logic"* systems considered in this paper are based on the systems described in [3], and [26]. The only difference is that now the logical OR and AND operators are interchanged. In this way we deal with CNF systems instead the DNF systems defined in [3].

The systems with id names: (see also Table I): 8A, 8B, 8C,8D, and 8E are defined on 8 atoms. Similarly, systems 16A, 16B, 16C, 16D, and 16E are defined on 16 atoms. Finally, systems 32A, 32C, and 32D are defined on 32 atoms. Systems 32B and 32E (described in [3]) were not considered due to excessive CPU requirements in obtaining computational results.

Each system was tested on 20 sequences of generating examples. For each such sequence initially ten examples are randomly generated and classified as either positive or negative by the expert (i.e., the *"hidden logic"*). Each example was a vector with *t* elements (*t* is the number of atoms). Each element was either 0 or 1 with probability 0.50. After an example was generated this way, it was classified by the *"hidden logic"* as either positive or negative.

Next, the OCAT algorithm was implemented to generate an initial version of $S_{GUIDED}$. What followed was the iterative generation of additional examples by the two different methods; GUIDED and RANDOM. Let $S_{RANDOM}$ be the Boolean function generated from the initial examples and the sequence of additional examples which were generated randomly (and $S_{GUIDED}$ is the Boolean function generated from the GUIDED input). In general, random examples in these experiments were generated as described above. In this way, after a sufficient number of random examples is classified by the *"hidden logic"*, the collection of the negative examples would be a representative sample of the total population of the negative examples of the *"hidden logic"*. The same issue is also true regarding the positive examples. Therefore, the computational experiments made no assumption regarding the distribution of the examples and the proposed guided learning strategy applies to any arbitrary distribution of the examples. After each pair of new examples was generated (one from GUIDED and one from RANDOM), the updated $S_{GUIDED}$ and $S_{RANDOM}$ systems were tested for convergence to $S_{HIDDEN}$.

Convergence of $S_{GUIDED}$ (or $S_{RANDOM}$) was assumed to have occurred if 10,000 randomly generated examples were classified correctly by $S_{GUIDED}$ (or $S_{RANDOM}$). This is admittedly an approximation, but our real interest is comparing the relative speed of convergence of $S_{GUIDED}$ and $S_{RANDOM}$ with $S_{HIDDEN}$. The comparison is simply how many additional examples are needed to correctly classify a random set of 10,000 observations.

Overall, the GUIDED strategy required on the average about 42% fewer examples than the RANDOM strategy for the entire set of 13 problems. The results for the individual problems are provided in Table I. Note for instance, that for problem 8A, the GUIDED strategy required on the average 34.85 examples to converge to a system equivalent to system 8A (used as the *"hidden logic"*). The same number under the RANDOM strategy is 59.55. Table I also presents the MIN, MAX, and standard deviations of the gathered observations. Figures 2.a - 2.d depict analytical results of the performance of the two strategies for four of the previous systems (selected randomly). These plots are in agreement with the summary results presented on Table I. It is also evident that strategy GUIDED outperformed, in almost all cases, the RANDOM strategy.

**Table I.**
Some Computational Results Under RANDOM and GUIDED Learning

| | **Number of Examples under RANDOM** | | | | **Number of Examples under GUIDED** | | | |
|---|---|---|---|---|---|---|---|---|
| System ID | MIN | Average | MAX | St.Dev. | MIN | Average | MAX | St.Dev. |
| 8A | 29 | 59.55 | 104 | 18.85 | 18 | 34.85 | 70 | 11.02 |
| 8B | 18 | 89.30 | 194 | 53.60 | 15 | 50.90 | 153 | 37.09 |
| 8C | 19 | 62.50 | 125 | 31.69 | 20 | 35.60 | 65 | 12.46 |
| 8D | 23 | 48.40 | 114 | 23.95 | 18 | 42.80 | 207 | 40.95 |
| 8E | 10 | 12.90 | 19 | 3.04 | 10 | 12.30 | 24 | 3.26 |
| 16A | 85 | 167.70 | 305 | 48.90 | 36 | 67.95 | 125 | 22.19 |
| 16B | 57 | 90.00 | 201 | 32.86 | 45 | 68.20 | 86 | 11.57 |
| 16C | 74 | 132.20 | 274 | 49.20 | 53 | 83.45 | 114 | 17.24 |
| 16D | 81 | 134.85 | 202 | 35.55 | 48 | 88.85 | 171 | 32.94 |
| 16E | 90 | 165.70 | 286 | 46.70 | 84 | 118.90 | 176 | 26.80 |
| 32A | 53 | 105.85 | 158 | 30.88 | 48 | 77.40 | 146 | 25.81 |
| 32C | 122 | 339.10 | 510 | 130.10 | 93 | 115.65 | 151 | 17.35 |
| 32D | 57 | 122.70 | 228 | 43.09 | 60 | 95.55 | 142 | 25.91 |

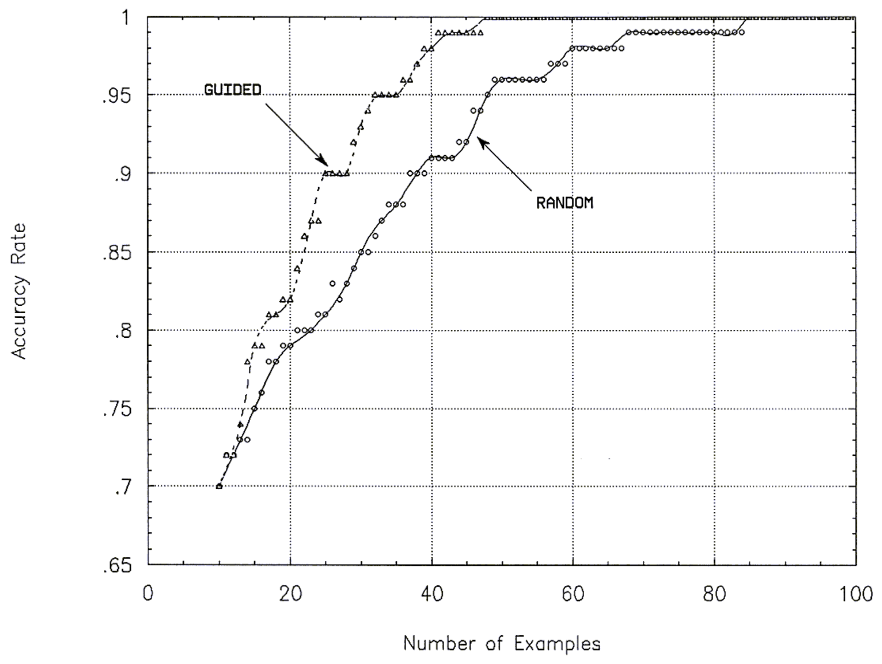

**Figure 2.a.**
Results When *"Hidden Logic"* is System 8A.

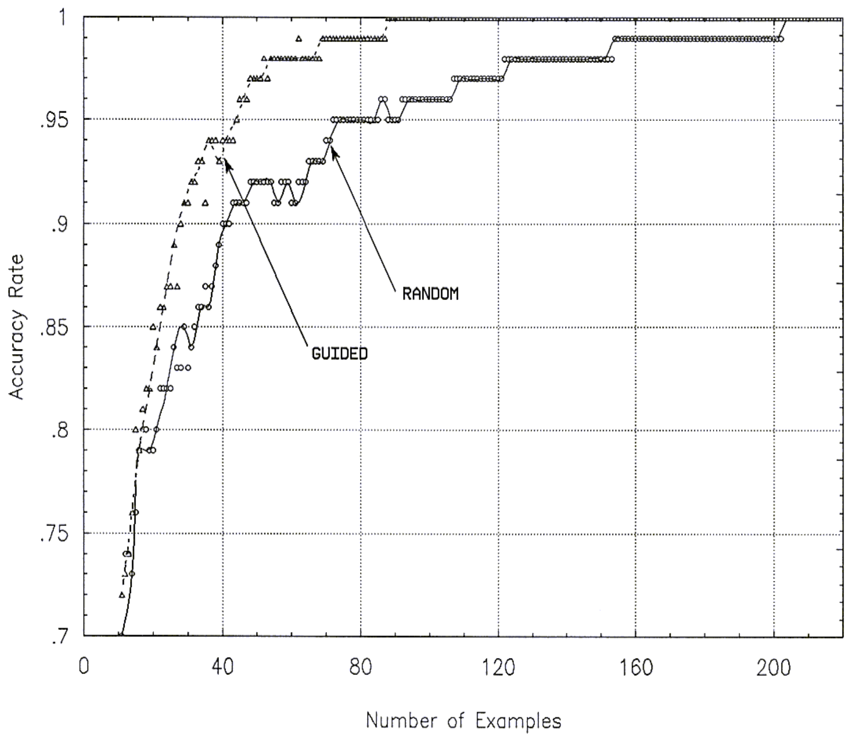

**Figure 2.b.** Results When *"Hidden Logic"* is System 16A.

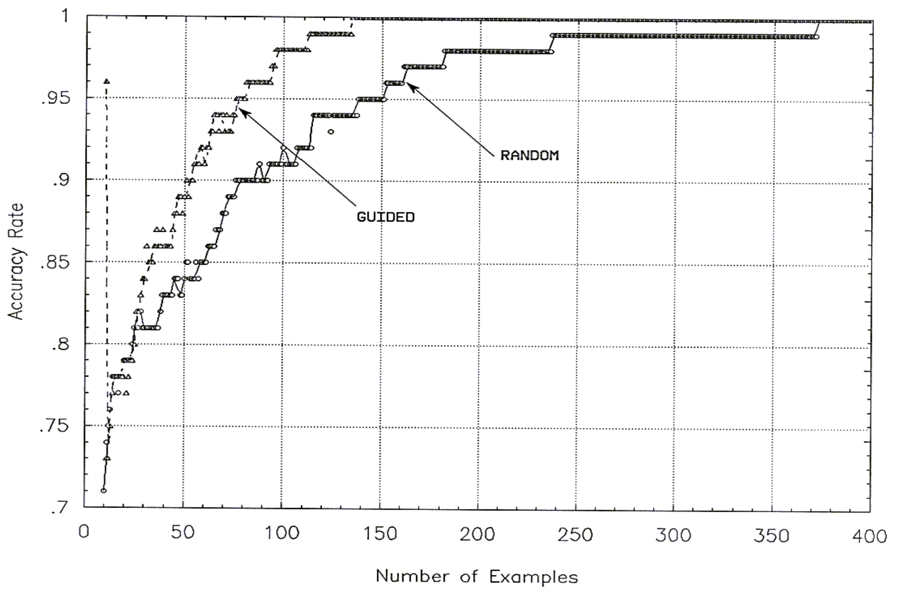

**Figure 2.c.** Results When *"Hidden Logic"* is System 32C.

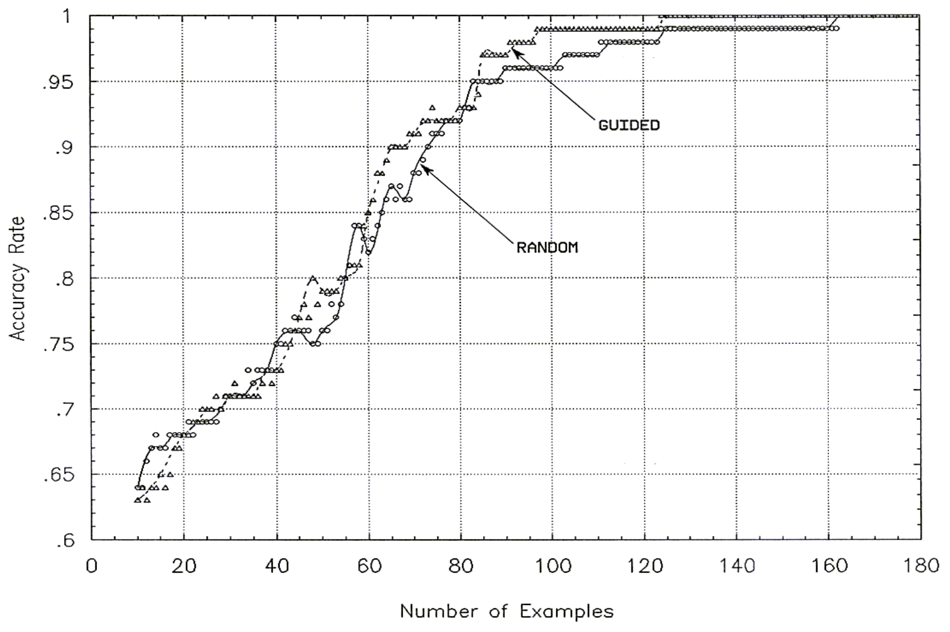

**Figure 2.d.**
Results When *"Hidden Logic"* is System 32D.

Figure 3 depicts what occurs as a single sequence of examples was generated for system 8B. This is a rather representative scenario. Note how $S_{GUIDED}$ uniformly dominates $S_{RANDOM}$. The bottom curve indicates how $S_{GUIDED}$ and $S_{R\text{-}GUIDED}$ become complements. As it can be seen from this figure, both strategies start from the same point (recall that **initially there are 10 random** examples). However, with the guided input strategy the inferred system reaches 100% accuracy much sooner (i.e., after 16 new examples, while with random input it takes 61 new examples).

Given a set of data, the derived system $S_{GUIDED}$ and a new example, the new proposed system may or may not be more accurate than its previous version. This depends on two factors: (1) on what is the approach used to infer the proposed system and (2) on the particular data. This is best described in the following illustrative example.

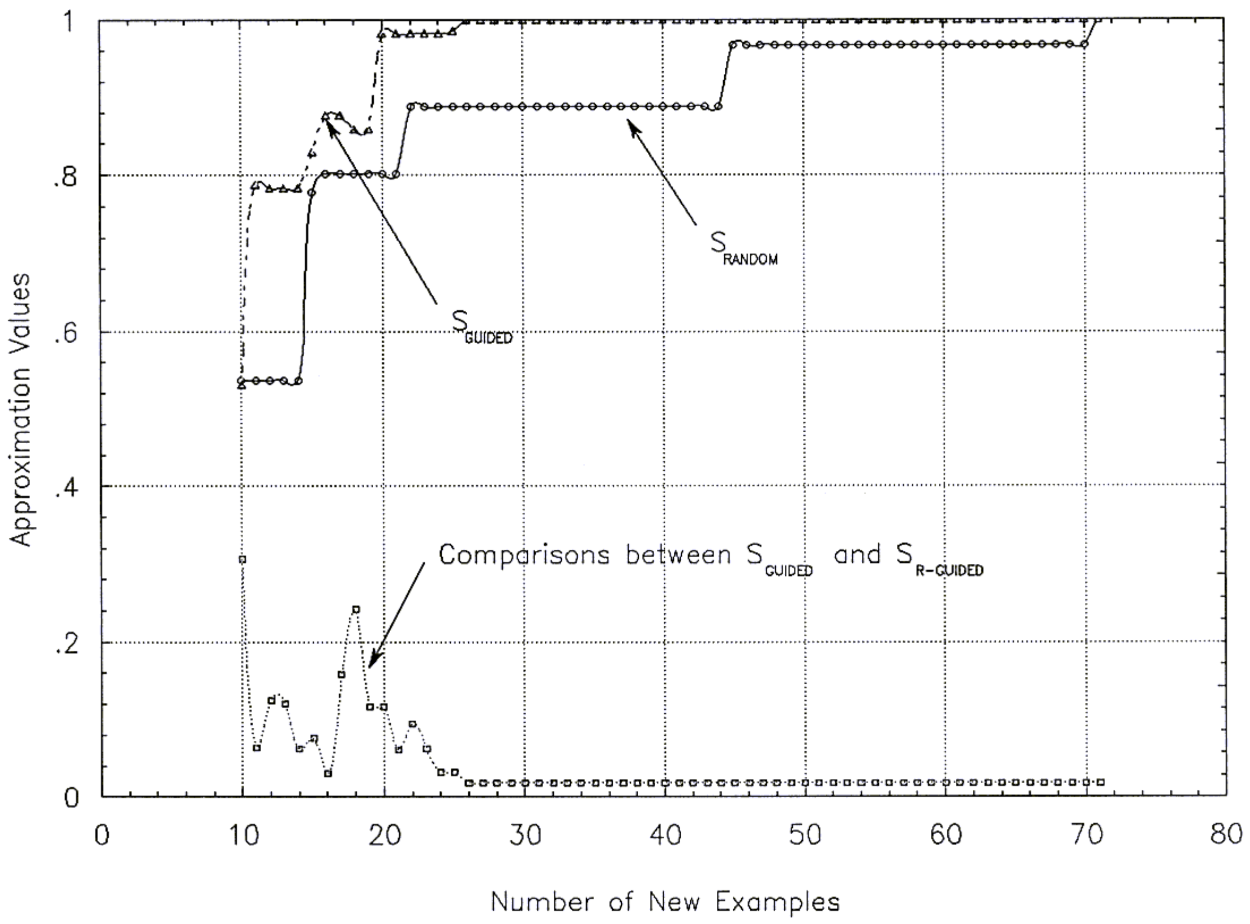

Figure 3. Comparisons between systems $S_{HIDDEN}$, $S_{GUIDED}$, and $S_{R-GUIDED}$ when new examples are considered ( system $S_{HIDDEN}$ is: $(\bar{A}_1 \vee \bar{A}_4 \vee A_6) \wedge (\bar{A}_2 \vee A_8) \wedge (A_2)$ ).

Suppose that the target function is: $(A_6 \vee \bar{A}_1 \vee \bar{A}_4) \wedge (A_8 \vee \bar{A}_2) \wedge (A_2)$ (i.e, system 8B). Let the two sets of positive and negative examples be as follows:

$$E^{+} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \qquad E^{-} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

When OCAT is used on the previous data, then $S_{GUIDED}$ is: $(A_2) \wedge (A_8) \wedge (A_3 \vee A_6 \vee A_7)$ and $S_{R\text{-}GUIDED}$ is: $(A_1 \vee \bar{A}_2 \vee \bar{A}_8) \wedge (A_5 \vee \bar{A}_2 \vee \bar{A}_3 \vee \bar{A}_8)$. When system $S_{GUIDED}$ is compared with the *"hidden logic"* system, then the derived accuracy is 0.95. Let (1 1 0 1 0 0 1 1) be the next example to consider.

It can easily be observed that this example is classified identically (e.g., as positive) by both the previous two systems. Next, this example is also classified by the *"hidden logic"*, and a contradiction between the proposed system and the *"hidden logic"* is revealed. Therefore, the set of negative examples is augmented by that example, and the OCAT approach is used again to derive the new version os system $S_{GUIDED}$. The updated system is:

$$(A_2) \wedge (\bar{A}_4 \vee \bar{A}_7) \wedge (A_3 \vee \bar{A}_1) .$$

The corresponding accuracy rate is 0.77, which is **smaller** than of the accuracy of the previous proposed system (although more examples are used as input). A similar situation is also depicted in Figure 3. In this figure the curve which corresponds to the guided strategy illustrates this phenomenon. Note that when the number of examples is 19, there is a **valley** in this curve.

**Table II.**

Computational Results when the Breast Cancer Data are Used

| % of Data Used for Training | with RANDOM Input | | with GUIDED Input | |
|---|---|---|---|---|
| | No. of Rules | Accuracy Rate | No. of Rules | Accuracy Rate |
| 10 | 1.6 | 0.88 | 1.63 | 0.87 |
| 15 | 2.06 | 0.88 | 2.26 | 0.91 |
| 20 | 2.42 | 0.89 | 3.09 | 0.92 |
| 25 | 2.72 | 0.9 | 3.91 | 0.94 |
| 30 | 3.12 | 0.9 | 4.43 | 0.95 |
| 35 | 3.4 | 0.9 | 5.29 | 0.97 |
| 40 | 3.74 | 0.9 | 5.94 | 0.97 |
| 45 | 4.06 | 0.9 | 6.77 | 0.98 |
| 50 | 4.44 | 0.9 | 7.23 | 0.98 |
| 55 | 4.9 | 0.9 | 7.74 | 0.99 |
| 60 | 5.4 | 0.9 | 8.11 | 0.99 |
| 65 | 5.84 | 0.91 | 8.11 | 0.99 |
| 70 | 6.24 | 0.91 | 8.14 | 0.99 |
| 75 | 6.92 | 0.91 | 8.17 | 0.99 |
| 80 | 7.68 | 0.91 | 8.34 | 0.99 |
| 85 | 7.88 | 0.91 | 8.6 | 0.99 |
| 90 | 8.56 | 0.91 | 8.77 | 0.99 |
| 95 | 8.88 | 0.92 | 8.83 | 0.99 |

It should be stated here that in these experiments no satisfiability formulation was used in the GUIDED strategy in order to accomplish the task of finding the next example. Instead, randomly generated examples were used to find an example which would be classified identically by the two systems $S_{GUIDED}$ and $S_{R\text{-}GUIDED}$. This was done for the sake of simplicity in order to keep CPU requirements low.

As it can be seen from the results in Table I, the guided input strategy was superior to random input strategy almost all the time. Only in the cases of systems 8E and 32E this was not the case. As it was anticipated, most of the time, the guided input learning strategy required considerably less examples in order to correctly infer a *"hidden logic"*.

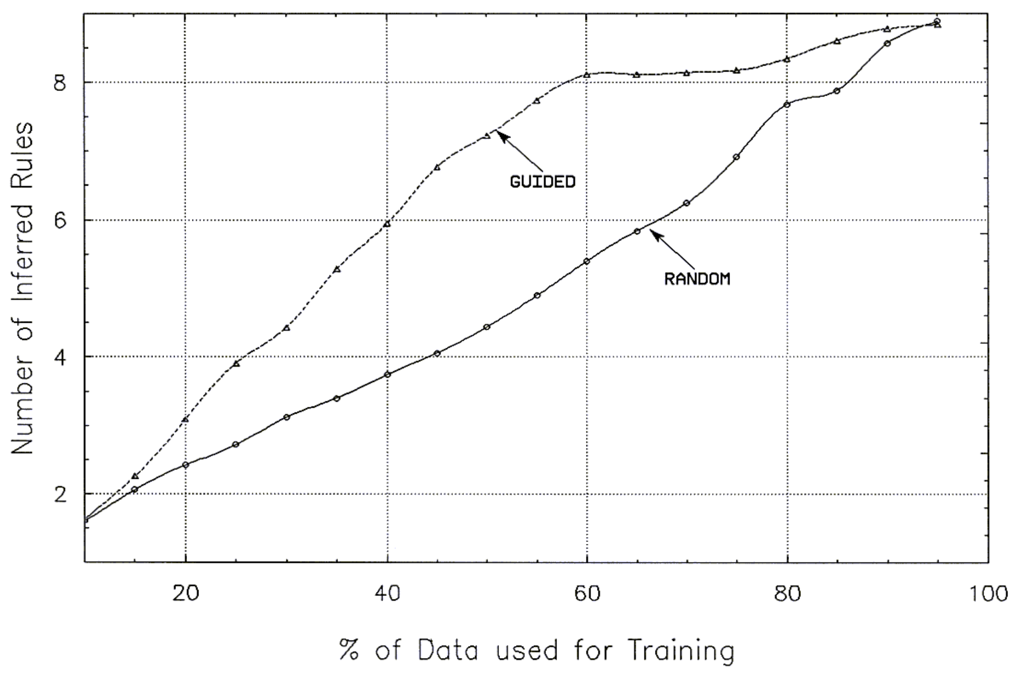

**Figure 4.a.** Results when the Breast Cancer Data are Used *(the focus is on the* ***Number of Rules****)*.

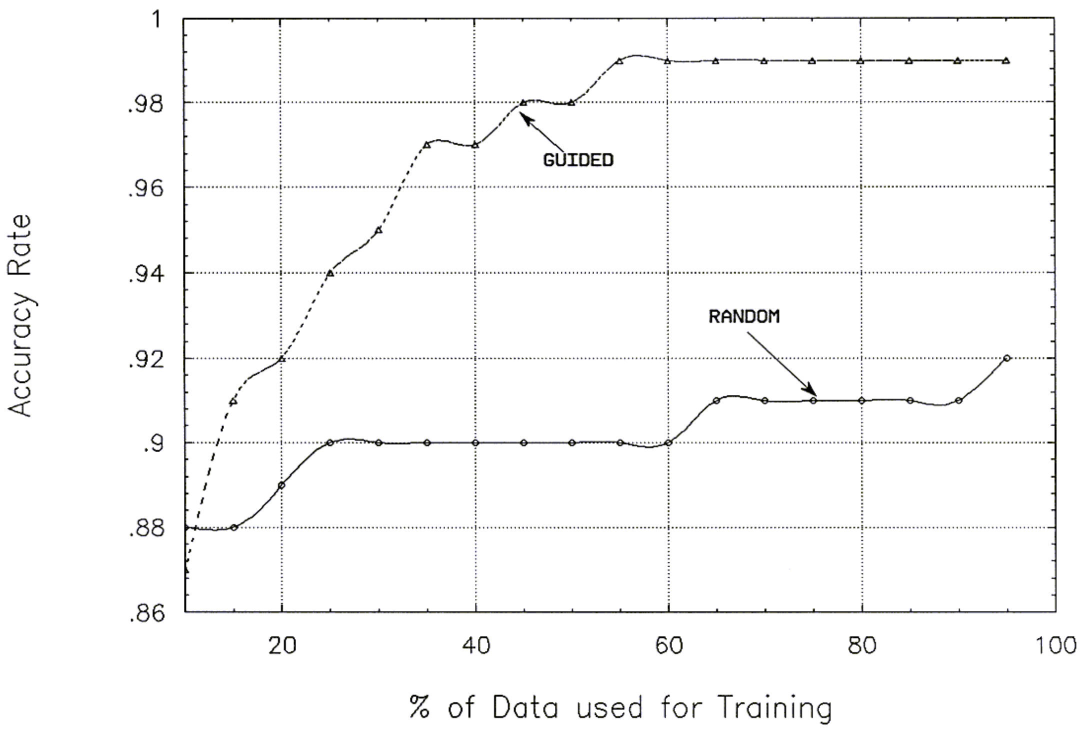

**Figure 4.b.** Results when the Breast Cancer Data are Used *(the focus is on the* ***Accuracy Rates****)*.

Professor Mangassarian from the University of Wisconsin and his associates have extensively used in their pattern recognition studies (see, for instance, [38]) a database with observations regarding nine cytological characteristics of breast tumors. Information for each tumor is derived by analyzing biological material extracted by using fine needle aspirates (FNAs). Each tumor was also classified as benign or malignant. At the time we performed our experiments were 421 cases, of which 224 were benign while the remaining 197 were malignant. These data are also available to the general public by anonymous *"ftp"* from the Machine Learning Database Repository at the University of California at Irvin, Department of Computer Science.

We transferred the data into the equivalent binary data and performed a series of computational experiments as follows. At first a 10% random collection of the original data was considered. Next, we generated the next example by using random input and also by using guided input (in two independent scenarios as before). That is, we applied an experimental procedure as with the previous experiments. However, now there is no *"hidden logic"* available, and thus we compared the accuracy of the derived systems in terms of how well they classified the **remaining** of the data (which were used as the testing data). For each experiment we used 50 random replications of it. The results of these tests are summarized in Table II and are also depicted in Figures 4.a and 4.b. In these results both the number of derived rules and the accuracy rates were recorded. As it can be easily verified from these results, once again the proposed guided strategy significantly outperformed the random input strategy. In these experiments as Boolean function inference algorithm we used the randomized heuristic described in [30] which is also based on the OCAT approach (as it best described in [25] and [26]). As a final comment it should be stated here that the accuracy rates in Figure 4.b did not necessarily reach the 100% value as the percent of the training data increased, because we were not comparing the inferred systems with a *"hidden logic"* but with the way they classified the **remaining** of the available data.

An interesting issue is to try to determine a way to conclude whether $S_{GUIDED}$ is a close approximation to $S_{HIDDEN}$. Intuitively, one expects that the closer the two systems $S_{GUIDED}$ and $S_{HIDDEN}$ become, the more apart the two systems $S_{GUIDED}$ and $S_{R\text{-}GUIDED}$ should become. One measure of the *"closeness"* of the two systems $S_{GUIDED}$ and $S_{HIDDEN}$ is determined by the percentage of 10,000 randomly generated examples which are classified identically by both systems.

This situation can be seen in Figure 3. Observe that in Figure 3 the systems $S_{GUIDED}$ and $S_{HIDDEN}$ become very close to each other (the top curve converges to value 1.00) when the bottom curve (which indicates the closeness of the systems $S_{GUIDED}$ and $S_{R\text{-}GUIDED}$) approaches the value 0.00. This is a rather representative case and other experiments demonstrated similar behavior.

This observation suggests the following **empirical test for system validation.** Suppose that one has generated a number of examples and has specified the two systems $S_{GUIDED}$ and $S_{R\text{-}GUIDED}$. Then, by using a sample of 10,000 randomly generated examples, the two systems $S_{GUIDED}$ and $S_{R\text{-}GUIDED}$ are compared in how often they agree in classifying these examples. If they agree very little (i.e., the approximation rate is very low), then it is very likely that the system $S_{GUIDED}$ is a **good approximation** of the corresponding *"hidden logic"*. However, if the approximation rate is very high, then it is rather **unlikely** that the system $S_{GUIDED}$ is an accurate approximation of the *"hidden logic"*.

## 8. CONCLUDING REMARKS

This paper discusses the development of a strategy for guided learning of Boolean functions from examples. The proposed method is based on the comparison of two Boolean functions (rule bases). The first function is the one derived from the positive and negative examples. The second function is derived by treating the positive examples as negative and the negative examples as positive. If it is possible to find a new example which is classified identically by both systems, then this example is considered next in the learning process. If no such example can be found, then the next example is generated randomly.

The computational results in this paper suggest that most of the time it is possible to find an example which is classified identically by both systems. In this way, the new example reveals that one of the two systems is inaccurate. Furthermore, the same computational results reveal that, on the average, the proposed strategy is significantly more effective than random learning. That is, most of the time in our computational experiments it was possible to infer a *"hidden logic"* much faster when the new examples were generated according to the proposed strategy, than when the examples were generated randomly.

An interesting issue for future research would be to expand the proposed methodology to more general situations. The present paper focused on the case dealing with propositional logic. There is no reason why the proposed methodology cannot be expanded to cover more general cases. From the previous discussions it follows that this methodology can be applied when it is possible to derive the two systems $S_{GUIDED}$ and $S_{R\text{-}GUIDED}$. Then, by determining as the next example an example which is classified identically by the two systems, the convergence of the proposed hypothesis to the target concept could be expedited. Clearly, more work is needed in this critical area of machine learning research.

**Acknowledgments**

The authors would like to thank Professor L.R. LaMotte, at Louisiana State University, for his valuable suggestions regarding the termination procedure of the proposed strategy.

## REFERENCES

1. J.G. Carbonell, R.S. Michalski, and T.M. Mitchell, An Overview of Machine Learning from Examples." R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach*, R.S. Tioga Publishing Company, Palo Alto, CA, 3-23 (1983).

2. T.C. Dietterich and R.S. Michalski, A Comparative Review of Selected Methods for Learning from Examples, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (eds.). *Machine Learning: An Artificial Intelligence Approach*, Tioga Publishing Company, Palo Alto, CA, 41-81 (1983).

3. A.P. Kamath, N.K. Karmakar, K.G. Ramakrishnan and M.G.C. Resende, A Continuous Approach to Inductive Inference, *Mathematical Programming*, **57**, 215-238 (1992).

4. M. Kearns, L. Ming, L. Pitt and L.G. Valiant, On the Learnability of Boolean Formulae, *J. of ACM*, **9**, 285-295 (1987).

5. L. Pitt and L.G. Valiant, Computational Limitations on Learning from Examples, *J. of ACM*, **35**(4), 965-984 (1988).

6. J.R. Quinlan, Induction of Decision Trees, *Machine Learning*, **1**, 81-106 (1986).

7. L.G. Valiant, A Theory of the Learnable, *Comm. of ACM*, **27** (11), 1134-1142 (1984).

8. C.E. Blair, R.G. Jeroslow, and J.K. Lowe, Some Results and Experiments in Programming Techniques for Propositional Logic, *Computers and Operations Research*, **13**, 633-645 (1985).

9. T.M. Cavalier, P.M. Pardalos, and A.L. Soyster, Modeling and Integer Programming Techniques Applied to Propositional Calculus, J.P. Ignizio (ed.), *Computers and Operations Research*, **17**(6), 561-570 (1990).

10. J.N. Hooker, Generalized Resolution and Cutting Planes, R.G. Jeroslow (ed.). *Annals of Operations Research*, **12**(1-4), 217-239 (1988a).

11. J.N. Hooker, A Quantitative Approach to Logical Inference, *Decision Support Systems*, **4**, 45-69 (1988b).

12. R.G. Jeroslow, Computation-Oriented Reductions of Predicate to Prepositional Logic, *Decision Support Systems*, North-Holland, **4**, 183-197 (1988).

13. R.G. Jeroslow, *Logic-Based Decision Support*, North-Holland, (1989).

14. A.P. Kamath, N.K. Karmakar, K.G. Ramakrishnan and M.G.C. Resende, Computational Experience with an Interior Point Algorithm on the Satisfiability Problem, *Annals of Operations Research*, P.M. Pardalos and J.B. Rosen (eds.). Special issue on: Computational Methods in Global Optimization, **25**,

43-58 (1990).

15. L.G. Valiant, Learning Disjunctions of Conjunctives, *Proceedings of the 9th IJCAI*, 560-566 (1984).

16. H.P. Williams, Linear and Integer Programming Applied to the Propositional Calculus, *International Journal of Systems Research and Information Science*, **2**, 81-100 (1987).

17. D. Angluin, Queries and Concept learning, *Machine Learning*, **2**, 319-342 (1988).

18. D. Haussler and M. Warmuth, The Probably Approximately Correct (PAC) and Other Learning Models. Chapter in: *Foundations of Knowledge Acquisition: Machine Learning*, A.L. Meyrowitz and S. Chipman (Eds), Kluwer Academic Publishers, Norwell, MA, pp. 291-312 (1993).

19. D. Angluin, Computational Learning Theory: Survey and Selected Bibliography, *Proceedings of the 24-th Annual ACM Symposium on the Theory of Computing*, Victoria, BC, Canada, May 4-6, 351-369 (1992).

20. D. Haussler, Learning conjunctive concepts in structural domains, *Machine Learning*, **4**, 7-40 (1989).

21. R.L. Rivest, Learning decision lists, *Machine Learning*, **2**(3), 229-246 (1987).

22. Y. Mansour, Learning of DNF Formulas, *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 53-59 (1992).

23. J. Peysakh, A Fast Algorithm to Convert Boolean Expressions into CNF, IBM Computer Science RC 12913 (#57971), Watson, NY (1987).

24. M. Bongard, *Pattern Recognition,* Spartan Books, New York, NY (1970).

25. E. Triantaphyllou, A.L. Soyster, and S.R.T. Kumara, Generating Logical Expressions from Positive and Negative Examples via a Branch-And-Bound Approach, *Computers and Operations Research*, **21**(2), 185-197 (1994).

26. E. Triantaphyllou, Inference of A Minimum Size Boolean Function From Examples by Using A New Efficient Branch-and-Bound Approach, *Journal of Global Optimization*, **5**(1), 69-94 (1994).

27. N.K. Karmakar, M.G.C. Resende and K.G. Ramakrishnan, An Interior Point Algorithm to Solve Computationally Difficult Set Covering Problems, *Mathematical Programming*, **52**, 597-618 (1991).

28. E. Triantaphyllou and A.L. Soyster, A Relationship Between CNF and DNF Systems Which are Derived from the Same Positive and Negative Examples, *ORSA Journal on Computing*, **7**(3), 283-285, (1995a).

29. E. Triantaphyllou, and A.L. Soyster, On The Minimum Number of Logical Clauses Which Can Be Inferred From Examples, submitted for publication to: *Computers and Operations Research*, (second revision) (1995b).

**UPDATED REFERENCE:**

E. Triantaphyllou, and A.L. Soyster, On The Minimum Number of Logical Clauses Which Can Be Inferred From Examples, *Computers and Operations Research*, **23**(8), 783-799, (1996).

30. A.S. Deshpande and E. Triantaphyllou, A Randomized Polynomial Heuristic for Inferring A Small Number of Logical Clauses from Examples, *Technical Report*, 18 pages, Department of Industrial and Manufacturing Systems Engineering, Louisiana State University, Baton Rouge, LA 70803-6409, (1995).

**UPDATED REFERENCE:**

A.S. Deshpande and E. Triantaphyllou, A Greedy Randomized Adaptive Search Procedure (GRASP) for Inferring Logical Clauses from Examples in Polynomial Time and Some Extensions, *Mathematical and Computer Modelling*, **27**(1), 75-99, (1998).

31. T. Mitchell, The need for biases in learning generalizations, Technical Report CBM-TR-117, Rutgers University, New Brunswick, NJ (1980).

32. B.K. Natarajan, On learning sets and functions, *Machine Learning*, **4**(1), 123-133 (1989).

33. V.N. Vapnik, *Estimating of Dependencies Based on Empirical Data*, Springer-Verlag, New York, NY, (1982).

34. D. Haussler, Quantifying inductive bias: AI learning algorithms and Valiant's learning framework, *Artificial Intelligence*, **36**, 177-221 (1988).

35. J. Shawe-Taylor, M. Antony, and N. Biggs, Bounding sample size with the Vapnik-Chervonenkis dimension, Technical Report CSD-TR-618, University of London, Surrey, England (1989).

36. A. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth, Learnability and the Vapnik-Chervonenkis dimension, *Journal of the Association of Computing Machinery*, **36** (4), 929-965 (1989).

37. P.E. Utgoff, ID5: An Incremental ID3, *Proceedings of the Fifth International Conference on Machine Learning*, John Laird (ed.). University of Michigan, Ann Arbor, Michigan, 123-132 (1988).

38. O. Mangassarian, R. Setiono and W.H. Woldberg, Pattern Recognition Via Linear Programming: Theory and Application to Medical Diagnosis. In: *Large-Scale Numerical Optimization*, Eds. T.F. Coleman, and Y. Li, SIAM, 22-30 (1991).