Learning from Examples vs. Programming by Demonstration Is Interaction the Key to (Better) Applications?

Lake Tahoe, USA, July 9th, 1995

Workshop at the Twelfth International Conference on Machine Learning

Proceedings

Proceedings Editor:

H. Friedrich (Karlsruhe, Germany)

MONOTONE BOOLEAN FUNCTION LEARNING TECHNIQUES INTEGRATED WITH USER INTERACTION

BORIS KOVALERCHUK[*], EVANGELOS TRIANTAPHYLLOU[*,1] AND EUGENE VITYAEV[**]

[*] *Department of Industrial Engineering, 3128 CEBA Building, Louisiana State University, Baton Rouge, LA 70803-6409, U.S.A. E-mail: borisk@unix1.sncc.lsu.edu, ietrian@lsuvm.sncc.lsu.edu*

[**] *Institute of Mathematics, Russian Academy of Science, Novosibirsk, 630090, Russia.*

Abstract. *This paper discusses some key issues about an interactive machine learning approach based on the theory of monotone Boolean functions. We present some key problems and the main steps of some algorithms for solving them. The concept of Shannon functions is used as a criterion for algorithmic optimality. The proposed approach allows for the possibility to decrease the number of positive and negative examples needed to infer a monotone Boolean function, and thus can make this type of machine learning systems more applicable.*

Key Words. Learning from Examples, Monotone Boolean Functions, Shannon Function.

1. INTRODUCTION

The general approach of pure machine learning and inductive inference includes the following two steps: (1) obtaining in advance a sufficient number of examples (vectors) for different classes of observations, and (2) formulation of the assumptions about the required mathematical structure of the example population (see, for instance, (Bongard, 1966), (Vityaev, 1975), (Zagoruiko, 1979), and (Vityaev and Moskvitin, 1993)). Human interaction is used just when we obtain examples and formulate the assumptions. In this paper we deal with monotone Boolean functions and we consider an extension of human interaction for the discrimination stage (i.e., when a monotone Boolean function is constructed from the examples). 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, finance as it is best demonstrated by the plethora of databases in the Machine Learning Repository in the University of California, at Irvine (Murphy and Aha, 1994).

The traditional machine learning approach has some difficulties, in particular for monotone Boolean functions. The size of the hypothesis space is influential in determining the **sample complexity** 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 (Mitchell, 1980), (Natarajan, 1989) and (Goldman and Sloan,1992). Usually the amount of bias in the hypothesis space H is measured in terms of the *Vapnik-Chervonenkis dimension*, denoted as *VCdim(H)* (Vapnik, 1982) and (Haussler, 1988). Theoretical results regarding the *VCdim(H)* are well known (Vapnik, 1982). The results in (Vapnik, 1982) are still better than some other bounds given in (Blumer, 1989). However, all these bounds are still overestimates (Haussler and Warmuth, 1993).

Before learning is carried out we do not know if the

1 *The first two authors gratefully acknowledge the support from the Office of Naval Research (ONR) grant N00014-95-1-0639.*

given examples are ***sufficient*** or ***surplus*** for successful learning. Often, the collected examples are less or more than the ones really needed. If we obtain inappropriate results after learning, we should try to add some examples and/or reject some of them as noise with the help of user interaction. Therefore, the real process of machine learning is an interactive process but it is not a formalized process yet. Interaction is not an element of the procedures for automated pure machine learning. Currently, user interaction is mostly considered as a requirement for user friendly software than the feature of machine learning algorithms (MLA).

In this paper we show that user interaction should be a point of improvement of MLA. Usual ML algorithms and software do not give any guidance on how to generate additional examples with user interaction. Moreover, often a generation of examples is very expensive and requires lots of time. Therefore, generation of surplus examples should be avoided and interactive ML algorithms, with the property of minimizing the number of required examples, should be developed (Schapire, 1992) and (Kearns et al.,1987).

Usually, interactive practice relies on some pragmatic assumptions, application of specific heuristics and very few examples. This approach has limitations on their adaptability. The purpose of this paper is to offer a more general approach to:

1) continually use human interaction for example generation;
2) construction of intuitively acceptable decision rules;
and 3) construction of concepts from examples.

The proposed approach is developed for Boolean vectors (positive and negative examples) and a sufficient general assumption of *"compactness of a pattern"*. We will show that the above assumption can be represented with a mathematical formalism of monotone Boolean functions. These functions have attracted attention in machine learning and regression analysis (see, for instance, (Boros et al., 1993), and (Kovalerchuk and Lavkov, 1984)). This happened because of their simplicity, the well developed mathematical theory and the clear interpretation they offer to many applications. Each discrimination (i.e., general Boolean function) can be described in terms of several monotone Boolean functions (see also the theorem in section 2). A small number of monotone functions allows for a simpler discrimination and *"compactness of patterns"*. Next, we describe the approach for the simplest case of inferring a single monotone Boolean function.

2. SOME BASIC DEFINITIONS AND RESULTS ABOUT MONOTONE BOOLEAN FUNCTIONS

Let E_n denote the set of all binary vectors of length n. Let α and β be two such vectors. Then, the vector $\alpha = (\alpha_1,\alpha_2,\alpha_3,\ldots,\alpha_n)$ precedes the vector $\beta = (\beta_1,\beta_2,\beta_3,\ldots,\beta_n)$ (denoted as: $\alpha \leq \beta$) if and only if the following is true: $\alpha_i \leq \beta_i$, for all $1 \leq i \leq n$. If, at the same time: $\alpha \neq \beta$, then it is said that α strictly precedes β (denoted as: $\alpha \prec \beta$). The two binary vectors α and β are said to be comparable if one of the relationships $\alpha \leq \beta$ or $\alpha \geq \beta$ holds.

A Boolean function $f(x)$ is monotone if for any vectors $\alpha,\beta \in E_n$, the relation $f(\alpha) \leq f(\beta)$ follows from the fact that $\alpha \leq \beta$. Let M_n be the set of all monotone Boolean functions defined on n variables. A binary vector α of length n is said to be the upper zero of a function $f(\alpha) \in M_n$, if $f(\alpha) = 0$ and, for any vector β such that $\beta \prec \alpha$, we have $f(\beta) = 0$. Also, we shall call the number of unities (i.e., the "*1*" elements) in vector α as its level and denote this by $U(\alpha)$.

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 (pre-complete) 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 (Alekseev, 1988).

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

$$\psi(n) = 2^{\binom{n}{\lfloor n/2 \rfloor}(1+\varepsilon(n))}$$

where $0 < \varepsilon(n) < c(\log n)/n$ and c is a constant (see, for instance, (Kleitman, 1969), and (Alekseev, 1988)).

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 $\alpha = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_n)$, yields the value of $f(\alpha)$. The problem posed here is that of finding all upper zeros of an arbitrary function $f \in M_n$ with the help of a certain number of accesses to the operator A_f. 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 (Korobkov, 1965):

$$\varphi(n) = \min_{F \in \mathcal{F}} \max_{f \in M_n} \varphi(F, f). \qquad (1)$$

The problem examined next is that of finding all upper zeros 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 (Hansel, 1966), that in the case of this problem the following relation is true (also known as *Hansel's lemma*):

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

Here $\lfloor n/2 \rfloor$ is the closest integer number to $n/2$ which is no more than $n/2$. In terms of machine learning the set of all upper zeros represents the <u>border elements</u> of the negative pattern. In an analogous manner we can also introduce the concept of lower units to represent the border of a positive pattern. In this since each monotone Boolean function represents "compact patterns".

A binary vector α of length n is said to be the <u>lower unit</u> of a function $f(\alpha) \in M_n$, if $f(\alpha) = 1$ and, for any vector β from E_n such that $\beta < \alpha$, we get $f(\beta) = 1$. We will also call the above defined monotone Boolean function to be an <u>increasing (isotone) monotone</u> Boolean function in contrast with a decreasing monotone Boolean function. A Boolean function is <u>decreasing (antitone) monotone</u>, if for any vectors $\alpha, \beta \in E_n$, the relation $f(\alpha) \leq f(\beta)$ follows from the fact that $\alpha \geq \beta$ (Rudeanu, 1974, p.149).

> *THEOREM: Each general discrimination (i.e., a general Boolean function) can be described in terms of several monotone Boolean functions.*

Proof:

It is well known that any Boolean function can be presented by its DNF. Also, each conjunction from a DNF representation can be presented by a pair of monotone Boolean functions. One of them is an increasing and the other one is a decreasing function.

Let $x_1 \wedge \ldots \wedge x_i \wedge \bar{x}_{i+1} \wedge \ldots \wedge \bar{x}_k$ be a conjunction from the DNF representation of a given Boolean function, where for simplicity the first i components (atoms) are positive while the next k-i components are negations. Then, we can form the Boolean functions

$$g(x_1, \ldots, x_n) = x_1 \wedge \ldots \wedge x_i$$

and

$$h(x_1, \ldots, x_n) = \bar{x}_{i+1} \wedge \ldots \wedge \bar{x}_k.$$

The function g is an increasing monotone Boolean function and the function h is a decreasing monotone Boolean function (Rudeanu, 1974). Hence, the conjunction $x_1 \wedge \ldots \wedge x_i \wedge \bar{x}_{i+1} \wedge \ldots \wedge \bar{x}_k$ is equal to the conjunction of functions g and h:

$$g(x_1, \ldots, x_n) \wedge h(x_1, \ldots, x_n).$$

Therefore, any arbitrary Boolean function q can be presented in the form:

$$q(x) = \bigvee_{j=1}^{m} \left(g_j(x) \wedge h_j(x) \right) \qquad (3)$$

where: $x = (x_1, \ldots, x_n)$ and m is an integer number. ■

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 (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, (Yablonskii, 1986) and (Alekseev, 1988)).

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.

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 just two compact monotone patterns. These conditions are natural for many applications. Such applications include the estimation of reliability (see also the illustrative example later in this section for problem 4).

Algorithm A1:
Step 1: The user is asked to confirm function monotony.
Step 2: Apply an iterative algorithm for generating examples and construction of the DNF representation. Do so by using Hansel's lemma (see also (Hansel, 1966) and (Gorbunov and Kovalerchuk, 1982)). This algorithm is optimal according to (1) and (2)).

PROBLEM 2.
Conditions:
There are no initial examples to initiate learning. There is a given binary vector (example) to be classified by the oracle A_f. Thus, we should learn to correctly classify a specific vector. However, now the direct interactive classification of a given example by A_f is associated with some kind of cost. Alternatively, the user may have an estimate of the probability $f(\alpha) = 1$. Also, the discriminant function should be a monotone Boolean function.

The basic idea of the proposed algorithm is follows. Let us suppose that vector α, (which should be classified) has the cost of classification C_α as determined by an oracle. We also suppose that we have at least two other significantly cheaper vectors, say α_* and α^*, with costs $C_{\alpha*}$, and C_α*, respectively (where: $C_\alpha >> C_{\alpha*}$, and $C_\alpha >> C_\alpha$*). These vectors should also satisfy the property: $\alpha_* \leq \alpha \leq \alpha^*$ (i.e., α is between them) and $f(\alpha_*) = f(\alpha^*)$ according to the oracle. Therefore, due to monotony, $f(\alpha)$ has the same value. In this way we can obtain the value of $f(\alpha)$ by exploiting monotony and a pair of low cost examples.

Algorithm A2:
Step 1: The user is asked to confirm function monotony.
Step 2: Estimation of the *"cost"* of any example which is considered by the user.
Step 3: Ordering the examples according to their *"cost"* values.
Step 4: Apply a cost related iterative algorithm for the generated examples.

Next, we present step 4 in more detail:

Step 4.1: Generate cheap examples which cover α (i.e., $\alpha_* \leq \alpha \leq \alpha^*$).
Step 4.2: Find examples α_* and α^* with the same answer of the oracle (i.e., $f(\alpha_*) = f(\alpha^*)$).

PROBLEM 3.

Conditions:
There are some initial examples for learning. 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). Also, the discriminant function should be a monotone Boolean function.

Algorithm A3:
Step 1: The user is asked to confirm function monotony.
Step 2: Test for monotony of the initial examples and reject the examples which violate the monotony property.
Step 3: Restoration of known elements of Hansel's chains (use: (Hansel, 1966), (Gorbunov, Kovalerchuk, 1982)).
Step 4: Apply a modified iterative algorithm for additional examples generation based on Hansel's lemma.

PROBLEM 4.

Conditions:
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). The discriminant functions should be monotone Boolean functions. There are some initial examples for learning for the connected classification problems simultaneously, with monotony supposition for both of them and the supposition that the positive patterns are nested.

Formally, the above consideration means that for all $\alpha \in E_n$, the following relation is true: $f_2(\alpha) \geq f_1(\alpha)$ (where $f_1(\alpha)$ and $f_2(\alpha)$ are the discriminant monotone

Boolean functions for the first and the second problems, respectively). The last situation is more complex than the previous one. However, the use of additional information from both problems allows for the potential to accelerate the rate of learning.

An Example of Nested Problems (A Reliability Example)

Let E^+_1 and E^-_1 be the sets of positive and negative examples for the first problem and E^+_2, E^-_2, be the sets of positive and negative examples for the second problem, respectively.

We consider the problem of classification of the states of some system. A qualified expert, working with the system for a long term can serve as an *"oracle"* (i.e., an operator A_f). States of the system are represented by binary vectors from E_n. The *"oracle"* 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 classification tasks: the first one for question 1 and the second task for question 2.

Task 1:

Pattern 1:	*"Guaranteed reliable states of the system"* (E^+_1).
Pattern 2:	*"Reliability of the states of the system is not guaranteed"* (E^-_1).

Task 2:

Pattern 1:	*"States of the system with some possibility for normal operation"* (E^+_2).
Pattern 2:	*"States of the system which guaranteed an accident"* (E^-_2).

In the above situation the following relations should be true: $E^+_2 \supset E^+_1$ and $f_2(\alpha) \geq f_1(\alpha)$ for all $\alpha \in E_n$, describing the system state, where $f_1(\alpha)$ and $f_2(\alpha)$ are discriminant monotone Boolean functions for the first and second tasks, respectively.

Algorithm A4:

Step 1: The user is asked to confirm the monotony for both tasks (i.e., the functions underlying their patterns).

Step 2: Testing of monotony of the initial examples for both tasks.

Step 3: Rejection of examples violating monotony.

Step 4: Restoration of $f(\alpha)$ values for elements of Hansel's chains, using monotony property and known examples (Hansel, 1966) and (Gorbunov and Kovalerchuk, 1982).

Step 5: Apply a dual iterative algorithm for additional examples generation based on Hansel's lemma.

Next, we discuss step 5 in more detail:

Step 5.1: Generate the next vector α_{i1} for interaction with A_{f1}.

Step 5.2: Generate the next vector α_{i2} for interaction with A_{f2}.

Step 5.3: Estimation of the numbers of vectors ($N^{1,1}_j$) for which we can compute $f_1(\alpha)$ and $f_2(\alpha)$ without asking oracles A_{f1} and A_{f2} by temporarily assuming that $f_j(\alpha_{i1}) = 1$ (for $j = 1,2$).

Step 5.4: Estimation of the numbers of vectors ($N^{1,0}_j$) for which we can compute $f_1(\alpha)$ and $f_2(\alpha)$ without asking oracles A_{f1} and A_{f2} by temporarily assuming that $f_j(\alpha_{i1}) = 0$ (for j = 1,2).

Step 5.5: Estimation of the numbers of vectors ($N^{2,1}_j$) for which we can compute $f_1(\alpha)$ and $f_2(\alpha)$ without asking oracles A_{f1} and A_{f2} by temporarily assuming that $f_j(\alpha_{i2}) = 1$ (for $j = 1,2$).

Step 5.6: Estimation of the numbers of vectors ($N^{2,0}_j$) for which we can compute $f_1(\alpha)$ and $f_2(\alpha)$ without asking oracles A_{f1} and A_{f2} by temporarily assuming that $f_j(\alpha_{i2}) = 0$ (for $j = 1,2$).

Step 5.7: Choose the variant with the maximal number of vectors from steps 5.3-5.6 to prefer α_1 or α_2 for asking the corresponding oracle.

Comment:
In step 5.7 we realize a local algorithm. This means that we consider just suppositions about $f_j(\alpha_{i1})$ and $f_j(\alpha_{i2})$. We do not simultaneously consider further suppositions about $f_j(\alpha_{i+1,1})$, $f_j(\alpha_{i+1,2})$, $f_j(\alpha_{i+2,1})f_j(\alpha_{i+2,2})$, etc. for the next possible results of interaction. This development of the algorithm can decrease the number of interactions, but leads to more computations.

4. BOOLEAN MACHINE LEARNING UNDER NOISE

PROBLEM DESCRIPTION

In this section we consider the case of noise in the data. There are many alternative formulations of the noise issue. In this paper we assume that a single example (i.e., binary vector) can be classified by both functions $f_n(x)$ (i.e., the *noise* function) and $f_h(x)$ (i.e., the target or *"hidden"* function). However, the analyst (user) does not have access to the values produced by the hidden function $f_h(x)$. That is, for a new example, say a, the user can retrieve the values $f_n(a)$, and $f_c(a) = f_n(a) \# f_h(a)$. In the last relation $f_c(a)$ is the composite value and "#" is a logical operator, which can be either assumed known or also to become subject of the investigation.

Therefore, the main problem is how to find the function $f_h(x)$ and the nature of the logical operator "#", given information on the outputs of the functions $f_n(x)$ and $f_c(x)$. From the above considerations it becomes apparent that the behavior of the target function $f_h(x)$ is obscured by the interaction with the second function $f_n(x)$, and thus the function $f_n(x)$ is called the noise function. For simplicity, in the following paragraphs we will assume that the logical operator "#" is the conjunction operator (i.e., &).

Next we present the highlights of some related algorithms which deal with noise. Algorithm NS1 assumes independent input to functions $f_c(x)$ and $f_n(x)$. Algorithm NS2 assumes that the same input is fed to both functions. Algorithms NS3, and NS4 deal with partially defined noise functions. Most of our algorithms deal with deterministic cases. For a stochastic setting the reader may want to consult with the work in (Schapire, 1992). Finally, algorithms NS5 and NS6 deal with cases in which there is probabilistic behavior of the noise function. A more detailed description of the main steps of these algorithms is given in below.

Algorithm NS1 (Parallel restoration of functions).

Step 1: Interactive restoration of the function $f_c(x)$.

Step 2: Interactive restoration of the function $f_n(x)$.

Step 3: Computational restoration of the function $f_h(x)$ on the base of: $f_c(x) = f_h(x) \& f_n(x)$, i.e.,

$f_h(x) = f_c(x) \& (\rceil f_n(x))$, where $\rceil f_n(x)$ indicates the negation of function $f_n(x)$.

Algorithm NS2 (Connected restoration of functions).

Step 1: Generation of the vector a for the oracles A_c and A_n.

Step 2: Obtaining $f_c(a)$ from the oracle A_c.

Step 3: Obtaining $f_n(a)$ from the oracle A_n

Step 4: Computational restoration of the value $f_h(a)$ on the base of $f_c(a)$ and $f_n(a)$, i.e., $f_h(a) = f_c(a) \& (\rceil f_n(a))$.

Step 5: Generation of the next vector a according to the Hansel lemma.

Step 6: Return to the steps 1-4 with each new vector until $f_h(x)$ will be restored.

Algorithm NS3 (Parallel restoration of functions with partially known an oracle for noise A_n).

Step 1: Interactive restoration of the function $f_c(x)$

Step 2: Interactive restoration of the function $f_n(x)$ as a partially known function.

Step 3: Computational restoration of the function $f_h(x)$ on the base of $f_c(x) = f_h(x) \& f_n(x)$, i.e., $f_c(x) \& (\rceil f_n(x)) = f_h(x)$ as a partially known function.

Step 4: Construction of the functions $f_{n*}(x) < f_n(x) < f_n^*(x)$ such that if $f_n(x)$ is not known for A_n then $f_{n*}(x)=0$ and $f_n^*(x)=1$.

Step 5: Construction of the functions $f_c(x) \& (\rceil f_{n*}(x))$ and $f_c(x) \& (\rceil f_n^*(x))$ as lower and upper estimates for $f_h(x)$.

Finally, observe that the less knowledge oracle A_n has, the lesser are the possibilities to avoid noise.

Algorithm NS4 (Connected restoration of functions with partially known the oracle A_n).
This algorithm is a combination of algorithms NS2 and NS3).

Algorithm NS5 (Parallel restoration of functions with partially known an oracle for noise A_n with probability distribution $P_n(x)$ of $f_n(x)$ for some x).

Step 1: Interactive restoration of the function $f_c(x)$

Step 2: Interactive restoration of the function $f_n(x)$ as a partially known function.

Step 3: Computational restoration of the function $f_h(x)$ on the base of $f_c(x) = f_h(x) \& f_n(x)$. i.e., $f_c(x) \& (\rceil f_n(x)) = f_h(x)$ as a partially known function.

Step 4: Construction of functions $f_{n*}(x) < f_n(x) < f_n^*(x)$ such that if $f_n(x)$ is not known for A_n then $f_{n*}(x)=0$ and $f_n^*(x)=1$.

Step 5: Construction of the functions $f_c(x) \& (\rceil f_{n*}(x))$ and $f_c(x) \& (\rceil f_n^*(x))$ as lower and upper estimates for $f_h(x)$.

Step 6: Obtaining probabilistic estimates $B_*(x)$ and $B^*(x)$ for the borders $f_c(x) \& (\rceil f_{n*}(x))$ and $f_c(x) \& (\rceil f_n^*(x))$, respectively.

Step 7: Choose of a preferred border:
If $B_*(x) > B^*(x)$, then f_h is estimated as $f_c(x) \& (\rceil f_{n*}(x))$;
If $B_*(x) < B^*(x)$, then f_h is estimated as $f_c(x) \& (\rceil f_n^*(x))$.

Algorithm NS6 (Connected restoration of functions with partially known an oracle for noise A_n with probability distribution $P_n(x)$ of $f_n(x)$ for some x. This algorithm is a combination of algorithms NS3 and NS5.

5. CONCLUDING REMARKS

Some computational experiments (see, for instance, (Gorbunov and Kovalerchuk, 1982) and (Triantaphyllou and Soyster, 1995)) have shown that it is possible to significantly decrease the number of questions for an oracle in comparison with the full number of questions (which is equal to 2^n) and also in comparison with guaranteed pessimistic estimation (formula (2) in section 2) for many functions from applications. Some close form results were also obtained for a connected problem of retrieval of maximal upper zero (Kovalerchuk and Lavkov, 1984). These results show that an interactive approach, based on monotone Boolean functions, has the potential to be very beneficial to interactive machine learning.

6. REFERENCES

1. Alekseev, V.B. (1988), *"Monotone Boolean Functions"*. Encyclopedia of Mathematics, v. 6, Kluwer Academic Publishers, 306-307.
2. Blumer A. Ehrenfeucht A. Haussler D. and Warmuth M.K. (1989) *Learnability and the Vapnik- Chervonenkis dimension*, Journal of the Association of Computing Machinery, 36 (4), 929-965.
3. Bongard, M. (1967), *Pattern Recognition*, Moscow, "Nauka" Publ. (in Russian, English translation, 1970, by Spartakos Press, NY).
4. Boros E., Hammer, P., and Hooker, J. (1993), *"Predicting cause-effect relationships from Incomplete Discrete Observations"*, Rutgers center for operations research, RUTCOR research report RRR 9-93, Rutgers University,USA.
5. Goldman and Sloan R.H.(1992) *The Power of Self-Directed Learning*, Machine Learning, Vol. 14, 271-294.
6. Gorbunov, Yu, and Kovalerchuk, B. (1982), *"An Interactive Method of Monotone Boolean Function Restoration"*, Journal of Academy of Science of UzSSR, Engineering, v. 2, 3-6 (in Russian).
7. Hansel, G. (1966), *"Sur le nombre des fonctions Boolenes monotones den variables"*. C.R. Acad. Sci. Paris, v. 262, n. 20, 1088-1090.
8. Haussler D. (1988) *Quantifying inductive bias: AI learning algorithms and Valiant's learning framework*, Artificial Intelligence, 36, 177-221.
9. Haussler D. and Warmuth M.(1993), *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, 291-312.
10. Kearns M., Li M., Pitt L., Valiant L. (1987) *On the learnability of Boolean formulae*. In Proceedings of the nineteenth annual ACM symposium on theory of computing, 285-295.
11. Kleitman, D. (1969), *"On Dedekind's problem: the number of monotone Boolean functions"*. Proc. Amer. Math. Soc. 21, 677-682.
12. Korobkov V.K. (1965) *On monotone Boolean functions of algebra logic*, In Problemy Cybernetiki , v.13, "Nauka" Publ., Moscow, 5-28 (in Russian).
13. Kovalerchuk, B., and Lavkov, V. (1984), *"Retrieval of the maximum upper zero for minimizing the number of attributes in regression analysis"*. USSR Computational Mathematics and Mathematical Physics, v. 24, n. 4, 170-175.
14. Mitchell T.(1980), *The need for biases in learning generalizations*, Technical Report CBM-TR-117, Rutgers University, New Brunswick, NJ.
15. Murphy P.M. and Aha D.W.(1994) *UCI repository of machine learning databases. Machine-readable data repository*, Irvine, CA,University of California, Department of Information and Computer Science.
16. Natarajan B.K. (1989), *On learning sets and functions*, Machine Learning, 4(1), 123-133.
17. Rudeanu, S., (1974), *"Boolean functions and equations"*, North-Holland, NY.
18. Triantaphyllou, E.(1994), *"Inference of a minimum size Boolean function examples by using a new efficient branch-and-bound approach"*. Journal of Global Optimization, v. 5, n. 1, 69-94.
19. Schapire R. (1992) *The design and analysis of efficient learning algorithms*. MIT Press.
20. Shawe-Taylor J. Antony M.and Biggs N.(1989), *Bounding sample size with the Vapnik-Chervonenkis dimension*, Technical Report CSD-TR-618, University of London, Surrey, England.
21. Triantaphyllou, E. and Soyster, A. (1995), *"An approach to guided learning of Boolean*

functions", to appear in: Mathematical and Computer Modeling.

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

23. Vityaev, E. (1975), *"An Algorithm of Inductive Inference"*, Computational Systems (Vychislitel'nye sistemy), Institute of Mathematics, USSR Academy of Science, Novosibirsk, n. 61, 28-36 (in Russian).

24. Vityaev, E., and Moskvitin, A.(1993), *"Introduction to discovery theory"*. Program system: DISCOVERY, Logical Methods in Informatics, Computational Systems (Vychislitel'nye sistemy), Institute of Mathematics, Russian Academy of Science, Novosibirsk, n. 148, 117-163 (in Russian).

25. Yablonskii, S. (1986), *"Introduction to discrete mathematics"*, Moscow, "Nauka" Publ. (in Russian)..

26. Zagoruiko, N. (1979), *"Empirical Forecast"*, "Nauka" Publ., Novosibirsk (in Russian).