PERGAMON

Computers & Operations Research 29 (2002) 1677–1700

computers & operations research

www.elsevier.com/locate/dsw

An incremental learning algorithm for constructing Boolean functions from positive and negative examples

Salvador Nieto Sanchez[a], Evangelos Triantaphyllou[a,*], Jianhua Chen[b], T. Warren Liao[a]

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

[b] *Department of Computer Science, Louisiana State University, 286 Coates Hall, Baton Rouge, LA 70803, USA*

Received 1 September 2000; received in revised form 1 February 2001

Abstract

This paper introduces an *incremental* algorithm for learning a Boolean function from examples. The functions are constructed in the disjunctive normal form (DNF) or the conjunctive normal form (CNF) and emphasis is placed in inferring functions with as few clauses as possible. This incremental algorithm can be combined with any existing algorithm that infers a Boolean function from examples. In this paper it is combined with the *one clause at a time* (OCAT) approach (Comput. Oper. Res. 21(2) (1994) 185) and (J. Global Optim. 5(1) (1994) 64) which is a non-incremental learning approach. An extensive computational study was undertaken to assess the performance characteristics of the new approach. As examples, we used binary vectors that represent text documents from different categories from the TIPSTER collection. The computational results indicate that the new algorithm is considerably more efficient and it derives more accurate Boolean functions. As it was anticipated, the Boolean functions (in DNF or CNF form) derived by the new algorithm are comprised by more clauses than the functions derived by the non-incremental approach.

Scope and purpose

There is a growing need for methods that can analyze information and infer patterns in a way that can be useful to the analyst. This is the core of data mining and knowledge discovery from databases. Such methods often infer a Boolean function from observations that belong to different classes. This paper presents a methodology that infers a Boolean function in an *incremental* setting. The proposed approach can be used in conjunction with existing algorithms that infer a Boolean function from

* Corresponding author. Tel.: +1-255-578-5372; fax: +1-255-578-5990.
E-mail address: trianta@lsu.edu (E. Triantaphyllou). http://www.imse.lsu.edu/vangelis/

0305-0548/02/$ - see front matter
PII: S0305-0548(01)00050-8

examples. The paper presents both algorithmic developments and also an extensive empirical study. The results presented in the paper suggest that the proposed incremental approach is highly promising.

Keywords: Incremental and non-incremental learning; Learning from examples; Boolean functions; DNF and CNF form; Machine learning; Data mining; The one clause at a time (OCAT) approach for inferring a Boolean function

1. Introduction

This paper introduces a new incremental learning from examples (ILE) algorithm for the inference of a Boolean function from examples. The derived functions are in disjunctive or conjunctive normal form (DNF or CNF, respectively) and emphasis is given in having as few DNF or CNF clauses (also known as "*terms*" in the literature) as possible. In this paper, the new algorithm is combined with an existing algorithm for non-incremental learning from examples (NILE) of Boolean functions from two disjoint collections of examples. However, the proposed incremental approach can be combined with *any* non-incremental (NILE) approach for deriving a Boolean function from examples. In this study, the NILE algorithm used with the proposed ILE approach is the OCAT (for *one clause at a time*) approach [1,2] which attempts to minimize (optimally or semi-optimally) the number of derived clauses. The OCAT approach is a data mining approach and has been studied and used in various application domains. An extensive description of the OCAT approach can be found in a special link from the homepage of the second author (i.e., http://www.imse.lsu.edu/vangelis/). Thus, in this paper the new approach will be called IOCAT, for *incremental* OCAT.

In order to assess the comparative value of the new approach vs. the old one (i.e., the non-incremental OCAT approach), we used examples derived by analyzing almost 3000 text documents from the TIPSTER collection of documents [3,4]. For this purpose, we used the document surrogate concept as introduced by Salton [5] in order to represent text documents as binary vectors. The TIPSTER collection is often used to evaluate information retrieval systems and machine learning algorithms. As classes for the training examples, we used documents from four document categories. These were documents related to the Department of Energy (DOE), Wall Street Journal (WSJ), Associated Press (AP), and technical documents from the ZIPFF collection. In addition, in order to define two disjoint classes, the following three class-pairs were formed: (DOE vs. ZIPFF), (AP vs. DOE), and (WSJ vs. ZIPFF). The various algorithms were compared in terms of three measures of performance as follows: (i) CPU time requirements, (ii) accuracy of the derived Boolean functions, and (iii) number of clauses in the derived Boolean functions.

This paper is organized as follows. Section 2 presents a formal description of the research problem studied in this paper. Section 3 briefly reviews the main parts of the related literature. Section 4 describes the proposed IOCAT (Incremental OCAT) algorithm. Sections 5 and 6 present and discuss the results of the computational study. The paper ends with a summary section.

2. Problem description

Suppose that collections of examples from two disjoint classes are somehow made available to a computerized classification system. Each example is a binary vector defined on n attributes (also known as Boolean variables or atoms). Each example comes with a class membership designation. Furthermore, this setting is assumed to be deterministic and no errors are considered. These two collections form the training examples. The task of a classification system is to analyze the information embedded in the training examples and infer a model that best captures the behavior of the hidden system. That is, we assume that there is a system that can classify these, and also more, examples. Thus, a main challenge is to use the available training examples to infer a Boolean function that in turn can be used to accurately classify new (and thus unclassified) examples. Therefore, the central problem studied in this paper can be summarized as follows:

Given are two disjoint collections of training examples. These collections represent two mutually exclusive classes of observations of the behavior of some hidden system of interest. Then the central problem is to use these collections of examples to extract a Boolean function that can best describe the behavior of the hidden system.

There are many different types of classification systems. Such systems are usually based in neural networks (NN), K-nearest neighbor, discriminant analysis, etc. In this study, we focus on classification systems that express the underlying model in the form of a Boolean function. Such a Boolean function is expressed in DNF or CNF form. The reason for focusing on a Boolean function is that often one desires to express the classification logic in terms of logical decision rules. That is, "IF... THEN..." type of logical statements. For the latter reason, an additional goal is to infer a Boolean function that is comprised of the minimal, or near minimal, number of DNF or CNF clauses (since such clauses can directly be transformed into logical decision rules).

For a given set of examples, the learned (inferred) Boolean function may not be an accurate representation of the hidden system. This is especially true if the two collections of the training examples are limited or they are not representative of the entire population of examples. The very next example may negate the current Boolean function, and thus it can initiate a revision of this function. In this paper it is also assumed that it is possible for the analyst to be able to define the structure of new (and thus unclassified) examples. That is, a new example can be sent to the hidden system (also known as the *classification expert* or *oracle*) for the determination of its class. This process is usually associated with some kind of cost. Thus, another main problem studied in this paper is how to decide which example to send for classification and include in the training set. The new example might be any one of the remaining unclassified binary vectors or may be one of a restricted subset of the remaining unclassified binary vectors.

This situation is a familiar problem in machine learning that is usually called the "Guided Learning" (GL) problem (see, for instance, [6–9]). One strategy in dealing with this problem is to randomly select the next example from the population of the remaining unclassified binary vectors. An alternative approach is to use an identification process that can determine

the composition of a new example in a way that ensures the currently inferred Boolean function to be modified. This is the main idea of the strategy proposed in [19]. That strategy will be the foundation of the new strategy proposed in this paper.

A methodological problem closely associated with guided learning is how to best modify an existing Boolean function when the classification of a new example reveals that the current Boolean function is inaccurate. A brutal force approach is to reconstruct the entire function from the beginning by using the entire sets of the training examples augmented with the new example. An alternative approach might be to repair only a few clauses of the existing Boolean function in a way that the modified function correctly classifies all the available training examples (i.e., the old training examples plus the new one that revealed the need for modifying the function). This is the main problem of interest in this paper.

3. Related developments from the literature

Fig. 1 shows two mutually exclusive sets of binary examples. The first set, denoted as E^+, represents the first class of training example and it is called the set with the positive examples. Similarly, the second set, denoted as E^-, represents the set with the negative training examples. All these examples are defined by the presence (i.e., "1" value) or absence (i.e., "0" value) of four attributes or atoms A_i (for $i=1,2,3,4$). A Boolean function, denoted as $\mathscr{F}$, that satisfies the requirements of these examples is also provided in Fig. 1.

By definition, the hidden system (Boolean function) accepts each positive example while it rejects each negative one. Consequently, the inferred Boolean function should also evaluate each positive example as true and each negative example as false. Later, such examples are used to evaluate the performance of the proposed approach on some large-scale simulated problems. These examples were defined by properly analyzing text documents. A text document can be considered as text defined over a finite set of keywords. Then, the presence or absence of a key word can be indicated with the zero-one value of a binary variable. The binary vectors formed this way are called "*document surrogates*" or just "*surrogates*" in the text analysis literature (see, for instance, [10,5,11,12]). In our tests, we used such examples that were defined on 800 binary variables and were extracted by analyzing a total of almost 3000 text documents (examples).

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

$$\mathscr{F} = (A_3 \vee \bar{A}_2) \wedge (\bar{A}_4 \vee A_2 \vee \bar{A}_1) \wedge (A_1 \vee \bar{A}_3)$$

Fig. 1. A sample training set of six positive examples and a set of four negative examples and a Boolean function implied by these data.

Extracting a Boolean function from collections of positive and negative examples (expressed as binary vectors) is an old problem. Some of the initial contributions are due to [13]. This is also the problem addressed with the use of Karnaugh maps [14]. Some recent developments can be found in [15]. This is an NP-complete problem (see, for instance, [16,17]). In such cases, the inferred Boolean function is usually expressed in DNF or CNF [18] as defined next as (1) and (2), respectively

$$\bigvee_{j=1}^{k}\left(\bigwedge_{i\in\rho_j} a_i\right) \tag{1}$$

and

$$\bigwedge_{j=1}^{k}\left(\bigvee_{i\in\rho_j} a_i\right). \tag{2}$$

In the previous expressions a_i is either A_i or $\bar{A}_i$. That is, a DNF expression is a disjunction of conjunctions (clauses or terms), and a CNF expression is a conjunction of disjunctions.

Let $\{A_1, A_2, A_3, \ldots, A_n\}$ represent a set of n Boolean variables or atoms. Also, let v be a binary vector that is defined on these n atoms. Furthermore, let $\mathscr{F}$ be a Boolean function that evaluates to either 0 or 1 depending on the combination of the values of the atoms in vector v. That is, $\mathscr{F}(v)=1$ or $\mathscr{F}(v)=0$, depending whether the vector is a positive or a negative example, respectively.

The OCAT approach [1,2] is a data mining approach. It infers a Boolean function by sequentially constructing one clause at each one of a finite number of iterations (hence the name "*one clause at a time*" or OCAT). The main steps are depicted in Fig. 2.

At the first iteration, the OCAT approach forms (for the CNF case) a clause that accepts all the training positive examples while it rejects as many training negative examples as possible. In the second iteration it constructs a clause that accepts all the positive examples and rejects as many of the remaining negative examples as possible. This process continues until a set of clauses is formed that accepts all the positive examples and collectively rejects all the negative examples (for the CNF case). In [20], a simple data transformation procedure is described that can be used with OCAT (or any algorithm that infers a Boolean function from examples) to infer a Boolean function in DNF or CNF. This is achieved by first complementing the vectors and then by treating the positive examples as negative and the negative examples as positive. When this is applied on an algorithm that produces a CNF (DNF) function, then the result will be a DNF (CNF) function and vice-versa.

The main step in the OCAT approach is Step 2 in Fig. 2. In [2], the inference of a single clause is achieved by using a rather effective and efficient branch-and-bound (B&B) approach. In the computational study reported in [2], the resulting Boolean functions were of minimal or near minimal size as far as the number of clauses is concerned. However, the B&B approach can be CPU time consuming when the size of the training sets is large. Thus, in [21], a polynomial time heuristic is proposed to deal with the task in Step 2.

This heuristic is described in Fig. 3. The proposed incremental OCAT (to be called IOCAT) approach is combined with this heuristic. Its structure is based on the definition of two evaluative functions termed $POS(a_i)$ and $NEG(a_i)$. The function $POS(a_i)$ returns the number of positive

Input: Training examples in two groups E^+ and E^-.
Output: A Boolean function (in CNF) that satisfies the requirements of the training examples.

Begin
$i = 0$; $C = \varnothing$; /* initializations */
do while ($E^+ \neq \varnothing$)
Step 1: Let $i \leftarrow i + 1$; /* i indicates the i-th iteration */
Step 2: Find a clause c_i which accepts all members of E^+ while it rejects as many members of E^- as possible;
Step 3: Let $E^-(c_i)$ be the set of the members of E^- which are rejected by c_i;
Step 4: Let $C \leftarrow C \cup c_i$;
Step 5: Let $E^- \leftarrow E^- - E^-(c_i)$;
repeat;

Fig. 2. The one clause at a time (OCAT) approach (for the CNF case).

Input: Training examples in two groups E^+ and E^-.
Output: A Boolean function (in CNF) that satisfies the requirements of the training examples.

$q = 0$; /* counter initialization */
do while ($E^- \neq \varnothing$)
$q \leftarrow q + 1$;
Let E^+ be the original set of positive examples;
$K_q = \varnothing$; /* initializing a clause */
do while ($E^+ \neq \varnothing$)
Step 1: Calculate the $POS(a_j)$ / $NEG(a_j)$ ratio for all atoms a_j;
Step 2: Choose the a_j according to the $max[POS(a_j) / NEG(a_j)]$ value;
Step 3: Let $K_q \leftarrow K_q \cup a_j$;
Step 4: Let $E^+(a_j)$ be the set of members of E^+ which are accepted when a_j is included in the current clause K_q;
Step 5: Let $E^+ \leftarrow E^+ - E^+(a_j)$;
repeat;
Let $E^-(K_q)$ be the set of members of E^- which are rejected by K_q;
Let $E^- \leftarrow E^- - E^-(K_q)$;
repeat;

Fig. 3. A fast heuristic for Step 2 of the OCAT approach (for the CNF case).

examples accepted in the current clause (in CNF) under construction if the atom a_i (where a_i is either A_i or $\bar{A}_i$) is included in the clause under construction. A similar meaning applies for the $NEG(a_i)$ function in regard with the negative examples. If $NEG(a_i) = 0$ in Step 2, then the atom a_j is given a very high priority for inclusion in the clause being formed [21]. That heuristic was

also combined with some randomization techniques and also with the previous B&B approach in order to deliver functions under different input sizes. Now the inferred functions are not of minimal or near minimal size, but their sizes are still small.

The algorithms for Step 2 in Fig. 2 (i.e., the B&B approach or the heuristic depicted in Fig. 3) infer a Boolean function from training examples even if a collection of training examples is updated by adding just one new example. This is known as NILE. This may be computationally expensive when one already has a Boolean function and then he/she needs to update it because a newly introduced single training example is misclassified by the current function. Extensive surveys of NILE learning can be found, for instance, in [22–29].

On the other hand, ILE may be an attractive strategy to modify an existing function when it misclassifies a newly introduced training example. Among the first contributions in ILE is the concept learning system (CLS) [22]. In CLS prior observations were selected at random and were replaced with new examples in order to reconstruct the new knowledge. The CLS approach was soon abandoned because the learning rates were slow. In [23], the AQ system [30] was adapted to learn incrementally by limiting the number of examples needed to reconstruct the faulty knowledge, which was expressed in the DNF form. The AQ system repaired this knowledge by using a Euclidean distance measure to identify new examples that were good concept representatives. Its goal was to reconstruct only those portions of the knowledge (i.e., a set of clauses that describe an individual concept) that caused the misclassification.

Later, Reine and Michalski [25] extended the AQ system into the GEM system that repairs only individual terms of a DNF expression. In the GEM system, only the faulty conjunctive terms were submitted to a generalization procedure along with the observations it currently covered and those that triggered the classification inconsistency. The results of this system suggested that (i) ILE methods may yield more complex concept descriptions than NILE methods, and (ii) knowledge updates might be less expensive using ILE than with the NILE methods.

Next, suppose that two sets of training examples have, somehow, become available. One set will be called the "*positive*" class and the other the "*negative*" class (these names are assigned arbitrarily). Then a function inference algorithm is applied on these training examples and a single Boolean function is learned. This Boolean function is derived in an attempt to infer the "*hidden*" system that classified these training examples. Since this function accepts all the positive examples while it rejects all the negative ones, we will call the function "*the set with the positive rules*" or just the "*positive rules*" (since a Boolean function in CNF or DNF can also be viewed as a set of rules). For convenience, this function will be denoted as R^+. Next, one can use the same Boolean function inference algorithm to construct the "*negative rules*" (to be denoted as R^-) by simply switching the roles of the training examples. That is, by treating the initial negative examples as the positive examples and vice-versa. Obviously, the negative rules (negative Boolean function) will reject all the positive examples while they will accept all the negative ones.

These two functions (i.e., the positive and the negative rules denoted as R^+ and R^-, respectively) can play a pivotal role in an incremental learning setting. This was first applied on the guided learning strategy described by Triantaphyllou and Soyster in [19]. In that strategy, the OCAT approach was used to infer the previous two Boolean functions in an incremental learning environment. That is, it was assumed that the analyst had control on determining the composition of the next example to be sent for classification to the oracle and then to be

included in the training examples. In the strategy described by Triantaphyllou and Soyster in [19], the next example selected was one that was classified (before sending it to the oracle for the actual class classification) *identically* by both functions. In this way, it was secured that after the actual class membership was determined, then one of the two functions will be modified and hopefully its classification accuracy would be improved. The empirical results reported in [19] strongly suggested that this guided learning strategy is superior to just randomly selecting the next example for inclusion in the two training sets.

The above issues are best formalized as follows. Suppose that the oracle classifies the new example. If the oracle classifies this new example as a positive one, then it will be denoted as e^+. Otherwise (i.e., if it is classified as a negative one), it will be denoted as e^-. When the new example is fed to the two Boolean functions R^+ and R^-, then one and only one of the following three scenarios is possible.

1. It has been classified *correctly* if and only if:
   (a) $R^+(e^+) = 1$ and $R^-(e^+) = 0$; or:
   (b) $R^+(e^-) = 0$ and $R^-(e^-) = 1$.
2. It has been classified *incorrectly* if and only if:
   (c) $R^+(e^+) = 0$ and $R^-(e^+) = 1$; or:
   (d) $R^+(e^-) = 1$ and $R^-(e^-) = 0$.
3. The new example triggers an *undecided* situation if and only if:
   (e) $R^+(e^+) = 1$ and $R^-(e^+) = 1$; or:
   (f) $R^+(e^-) = 1$ and $R^-(e^-) = 1$; or:
   (g) $R^+(e^+) = 0$ and $R^-(e^+) = 0$; or:
   (h) $R^+(e^-) = 0$ and $R^-(e^-) = 0$.

Therefore, in the proposed ILE approach, the new examples will be determined such that scenario 3, above, occurs as frequently as possible. Such an example can be determined by solving a SAT (clause satisfiability) problem or by simply randomly sampling a large enough sample of unclassified examples until an example that is classified identically by both Boolean functions is detected [19].

4. The proposed incremental algorithm

The proposed incremental learning algorithm has some similarity to the GEM system [25]. They are similar in the sense that any disagreement between the inferred system and the training examples is not allowed and only the disjunctive terms triggering the wrong classification are repaired in the proposed algorithm. Nonetheless, they differ in the way new training examples are selected and used to reconstruct the current system. For instance, in the GEM system new information is submitted to a generalization process only when a set of misclassified examples has been collected. In contrast, in this paper this knowledge (i.e., the group of the two Boolean functions) is repaired by considering examples identified as "*undecided*". This guarantees either one of the positive or the negative Boolean function to be altered. Furthermore, the approach presented here differs from the GEM system because we always maintain two Boolean functions as described earlier.

In the GEM system, the methodology for repairing only the portion(s) of the function followed the procedures described in [23]. In this paper, however, this repair was divided into the following two mutually exclusive sub-problems that capture all possibilities: (i) Repair of a Boolean function that incorrectly rejects a positive example, and (ii) repair of a Boolean function that incorrectly accepts a negative example. For both sub-problems, we assume that the inferred Boolean function is in DNF. The CNF case can be developed in a similar manner. However, it seems that for this kind of problems the DNF case is more intuitive to follow.

4.1. Repairing a Boolean function that incorrectly rejects a positive example

From the definition of the DNF form given as expression (1) in Section 3, a Boolean function $\mathscr{F}$ accepts (i.e., it evaluates to true) an example if and only if at least one of its disjunctive clauses (terms) accepts it. Alternatively, a Boolean function rejects an example if and only if all its clauses (terms) reject it. Next, suppose that the current system, denoted as Boolean function $\mathscr{F}$, incorrectly rejects the positive example e^+. Then, the following relation (3) should be obviously satisfied:

$$\mathscr{F}(e^+) = c_1 \vee c_2 \vee c_3 \vee \cdots \vee c_n = 0, \tag{3}$$

where c_i (for $i = 1, 2, 3, \ldots, n$) is the ith clause (term) of $\mathscr{F}$.

The previous discussion naturally raises the question of how to decide the clause, among the n clauses c_i (for $i = 1, 2, 3, \ldots, n$) in relationship (3), one should alter such that the new positive example will be accepted by the modified Boolean function. The algorithm depicted in Fig. 4 addresses this problem.

This algorithm indicates (in Step 3) that two extreme strategies can be implemented. The first strategy is to select for change (repair) the clause that is the most generalizing clause (denoted as the MGC clause), while the second strategy is to select for repair the least generalizing clause (denoted as the LGC clause) in Fig. 4. These two strategies represent two extreme scenarios. They rank the clauses according to their generalization capability and then select the two extreme cases. In this way, it is hoped that one can study all possibilities. The generalizability of a clause is assessed in term of two parameters. One is the number of positive examples accepted by that clause. This is represented as $|E^+(c_i)|$ in Fig. 4. The higher this number is, the higher the generalizability of the clauses is assumed to be. The second parameter is the size of the clause denoted as $A(c_i)$. As size, we consider its length or the number of the atoms that define it. The fewer the atoms, the more general the clause is (for the DNF case). For these reasons, the most generalizing clause (MGC) in Fig. 4 is defined as the clause with the maximum $|E^+(c_i)|/A(c_i)$ value. Similarly, the least generalizing clause (LGC) is the one that corresponds to the minimum $|E^+(c_i)|/A(c_i)$ value.

It should be stated at this point that if one ranks the clauses of Boolean function according to their generalizability power, then one may expect that the clause that ranks the highest (i.e., the MGC clause), has also the most potential to effect the behavior of the Boolean function when that clause is altered. After all, by definition a LGC clause plays a smaller role. Therefore, it is reasonable to expect that by focusing the attention on the MGC will lead to better results. However, this is not possible to assess quantitatively without some kind of computational

Input:
The training sets E^+ and E^-. A Boolean function $\mathcal{F}$ in DNF that accepts all the positive examples while it rejects all negative ones. This function is comprised of n clauses (in DNF) denoted as c_i (for $i = 1, 2, 3, ..., n$). A new positive example that is incorrectly rejected by $\mathcal{F}$.

Output: A modified Boolean function $\mathcal{F}'$ (in DNF) that accepts all positive examples in $E^+ \cup \{e^+\}$ and it rejects all negative examples in E^-.

begin

Step 1: Let $E^+(c_i)$ (for $i = 1, 2, 3, ..., n$) be the set of the members of E^+ which are accepted by clause c_i ;

Step 2: Let $A(c_i)$ be the number of atoms in c_i;

Step 3: Select a clause c_k (for some k; $1 \leq k \leq n$) according to a clause selection criterion (i.e., MGC or LGC, as described below);

Step 4: Let $\mathcal{F} \leftarrow \mathcal{F} - c_k$;

Step 5: Let $E^+(c_k) \leftarrow E^+(c_k) \cup \{e^+\}$;

Step 6: Let f be the function (in DNF) that solves the sub-problem OCAT($E^+(c_k)$, E^-);

Step 7: Set $\mathcal{F}' \leftarrow \mathcal{F} \cup f$;

end;

Clause Selection Criteria:

1. Most Generalizing Clause (MGC): A clause c_i with the max $\{|E^+(c_i)| / A(c_i)\}$ value.
2. Least Generalizing Clause (LGC): A clause c_i with the min $\{|E^+(c_i)| / A(c_i)\}$ value.

Where $|x|$ indicates the cardinality of set x.

Fig. 4. Proposed strategy for repairing a Boolean function which incorrectly rejects a positive example (for the DNF case).

experiments. The computational results reported in the next section indicate that this hypothesis seems to be indeed the case.

In the same figure, the notation OCAT ($E^+(c_k)$, E^-) denotes a Boolean function inference problem that has as positive examples the members of the set $E^+(c_k)$ and as negative examples the members of the set E^-. This notation is used in the remaining of this paper. The effectiveness of these two selection criteria is further studied empirically later in terms of the size and the accuracy of the produced Boolean functions and also the required CPU times.

A key step for achieving an effective and efficient incremental learning solution is the size of the sub-problem OCAT($E^+(c_k), E^-$) in Step 6 of the algorithm depicted in Fig. 4. This is a key concept because if it is assumed that $|E^+(c_k)| \ll |E^+|$ (where $|x|$ is the size of the set x), then it is reasonable to assume that the CPU time for solving the sub-problem OCAT($E^+(c_k), E^-$) would be significantly shorter than the time required to solve the complete (and much bigger) problem OCAT($E^+ \cup \{e^+\}, E^-$). The branch-and-bound (B&B) approach that is used to solve such a problem in [2] deals with an NP-complete problem. The faster heuristic described in [21] is of polynomial time complexity. At this point, it is also important to notice that by using

Input: The negative example e^- that is incorrectly accepted by the function $\mathscr{F} = c_1 \vee c_2 \vee \ldots \vee c_n$ (in DNF). The two training sets E^+ and E^-.

Output: A Boolean function $\mathscr{F}^{//}$ that accepts all examples in E^+ and rejects all examples in $E^- \cup e^-$.

begin

Step 1: Let C be the set of clauses c_i (for $i = 1, 2, 3, \ldots, n$) that incorrectly accept e^-;

Step 2: Let $\mathscr{F} \leftarrow \mathscr{F} - C$;

Step 3: Let $E^+(C)$ be set of the members of E^+ which are accepted by C;

Step 4: Let f be the Boolean function in DNF form that solves the sub-problem OCAT($E^+(C)$, $E^- \cup \{e^-\}$);

Step 5: Let $\mathscr{F}^{//} \leftarrow \mathscr{F} \cup f$;

end;

Fig. 5. Repair of a Boolean function that erroneously accepts a negative example (for the DNF case).

either one of the two clause selection criteria, it may happen that one or more clauses might be added to the function $\mathscr{F}$ increasing this way its size. According to [23], this situation can be anticipated because the utilization of an ILE approach often results in more complex systems.

4.2. Repair of a Boolean function that incorrectly accepts a negative example

The algorithm in Fig. 5 addresses the second scenario for repairing a Boolean function $\mathscr{F}$ (in DNF). This scenario occurs when the Boolean function incorrectly accepts as positive a new example that has been classified by the oracle as negative (recall that this example is now denoted as e^-). In this case the current function $\mathscr{F}$ erroneously satisfies the condition:

$$\mathscr{F}(e^-) = c_1 \vee c_2 \vee c_3 \vee \cdots \vee c_n = 1. \tag{4}$$

The function in (4) shows that at least one of the n clauses incorrectly accepts the negative example e^-. The main problem in this scenario is how to select the clause(s) to repair so that the updated function, denoted as $\mathscr{F}^{||}$, will reject the negative example e^- while maintaining the correctness for the other examples (positive and negative).

The algorithm in Fig. 5 solves the problem implied in relation (4) by first identifying the set of clauses C that incorrectly accept the negative example e^-, and then by forming a subset of positive examples (denoted as $E^+(C)$), which is comprised of the examples in E^+ that are accepted by the clauses in C. The set of negative examples is formed by $E^- \cup \{e^-\}$. As with the sub-problem in the first scenario, the potential of the algorithm in Fig. 5 is based on solving the smaller size Boolean function inference problem OCAT($E^+(C), E^- \cup \{e^-\}$) in Step 4. This is an important issue because if $|E^+(C)| \ll |E^+|$ (where $|x|$ is the size of the set x), then the CPU time requirement for solving this sub-problem is most likely significantly shorter than that for solving the entire Boolean function inference problem OCAT($E^+, E^- \cup \{e^-\}$).

4.3. Computational complexity of the algorithms for the ILE approach

An inspection of the algorithms in Figs. 4 and 5 indicates that in the worst case, the entire problems OCAT($E^+ \cup \{e^+\}, E^-$) and OCAT($E^+, E^- \cup \{e^-\}$) will have to be executed. It should be emphasized here that the Boolean function inference problems described so far can be solved with *any* function construction method and not only the OCAT approach. In the experiments described in the next section these problems were solved by using the fast heuristic of polynomial time described in [21]. Another alternative would be to use the B&B approach described in [2]. Other methods could be used as well. The heuristic described in [21] is of $O(mn^2)$ time complexity (where m is the total number of examples and n is the number of binary attributes). On the other hand, the B&B approach is an NP-complete problem. Thus, the proposed incremental learning approach takes the time complexity of the learning algorithm used to solve the small function inference sub-problems.

The next section describes an extensive empirical study of the relative effectiveness of the ILE and NILE approaches when they are combined with the fast heuristic described in [21] for inferring a Boolean function. This empirical study examines the relative performance of the two clause-selection criteria described in Fig. 4 (i.e., the MGC and LGM criteria). As measurements of performance we used the CPU time, accuracy of derived systems (i.e., how accurately they classified the remaining available examples) and the size of the derived systems. The various approaches were analyzed in terms of the *sign test* [31] in order to determine any difference in the performance of pairs of algorithms. Please note that the sign test is a non-parametric test that compares paired observations of two populations. The number of positive and negative signs of the comparisons is used to make inference on the two populations of observations.

5. Experimental data

Table 1 shows the numbers of documents from the TIPSTER collection that were used in the experimentation. The TIPSTER collection is comprised of numerous documents extracted from various sources. As it was mentioned in Section 1, this collection of documents is often used to evaluate the performance of information retrieval (IR) and machine learning systems. These documents were randomly extracted from the four classes of the collection. The numbers in each class were determined from the RAM limitations of the PC we used in the experiments. The computer used was a Pentium II PC with a 400 MHz CPU running the Windows 95 OS. The computer programs for this study were written in Turbo Pascal 1.5 for Windows [32].

In order to simulate two mutually exclusive classes, the following three class-pairs (DOE vs. ZIPFF), (AP vs. DOE), and (WSJ vs. ZIPFF) were formed. These three class-pairs were randomly selected from all possible pair combinations. Furthermore, to comply with the notation presented in earlier sections, the first class of each class-pair was denoted as E^+, while the second class was denoted as E^- (these class designations were set randomly). The conversion of these documents into binary vectors followed the methodology discussed in [5,11,12,33–35]. It should be mentioned here that similar examples, also derived from large collections

Table 1
Number of documents randomly extracted from each class[a]

Class	DOE	AP	WSJ	ZIPFF	Total
Number of documents	1407	336	624	530	2897

[a]DOE, AP, and WSJ stand for Department of Energy, Associated Press, and the Wall Street Journal, respectively; ZIPFF is a collection of technical documents of various topics.

of text documents, were used in a study reported in [36]. In that study the OCAT approach was compared with the vector space model (VSM) [5] that is the traditional method used for document classification. It is noticeable that in that study, the OCAT approach significantly outperformed the VSM method.

6. Analysis of the computational results

The computational experiments were conducted as follows. First, a collection of examples was formed under one of the target class-pairs as defined in the previous section (Table 1). The test examples were derived by analyzing the text documents in the previous TIPSTER categories. The size of each example was detrimental by reaching the limits depicted in Table 1 for each class-pair. Next, an example was retrieved from the class-pair collection and was presented to the learning algorithm, along with its actual class membership. We used three learning algorithms as follows. The first one was the OCAT approach with the polynomial time complexity heuristic described in Fig. 3. The second algorithm is the ILE approach coupled with the MGC selection criterion and also the heuristic described in Fig. 3. The third ILE algorithm was similar to the second one, but now the LGC selection criterion is used instead of the MGC one.

The results are depicted in Figs. 6–14. In these figures the thickest lines correspond to results under the plain OCAT approach, the thick lines to results under the IOCAT when it is combined with the MGC criterion, while the thin lines to results under the IOCAT when it is combined with the LGC criterion. In the horizontal axes the term "documents" is used instead of "examples", since examples correspond to documents from the TIPSTER collection.

The results are grouped into three sub-sections with three figures in each sub-section. We present the plots for each class pair individually, in order to maintain some subtle differences that were observed in these results. The first sub-section deals with the accuracy of the derived system (i.e., the combinations of the "*positive*" and "*negative*" Boolean functions). The second sub-section deals with the number of clauses in the derived Boolean functions, while the third sub-section deals with the CPU time required by each approach.

6.1. Results on the classification accuracy

In these experiments the different learning processes started with the same initial random collection of 50 examples and then proceeded by incrementing the training examples one at a time according to the corresponding methods. The accuracy was defined as the number of

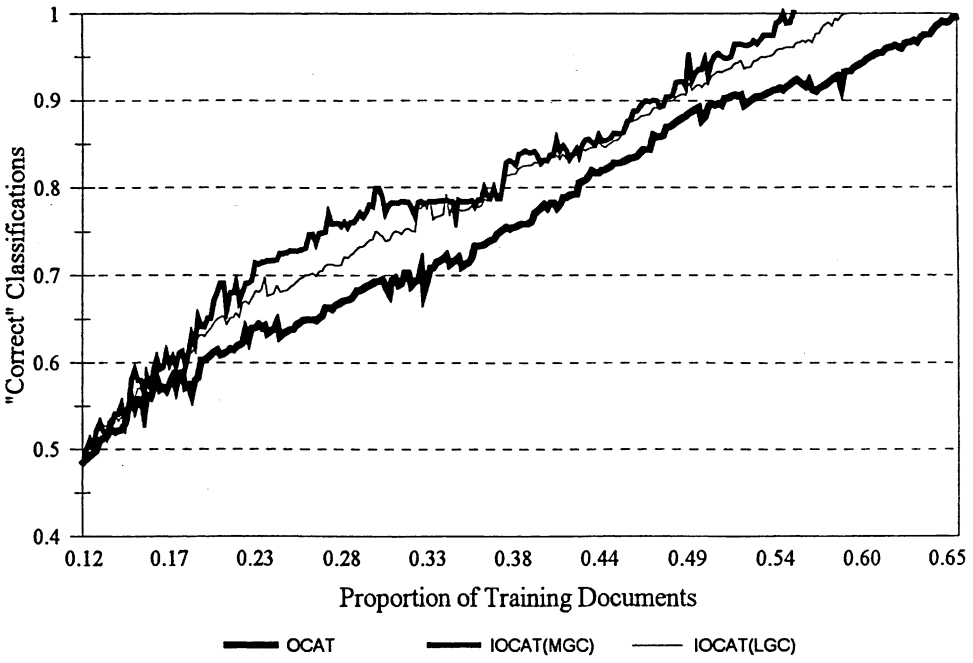

Fig. 6. Accuracy results for the class-pair (DOE vs. ZIPFF).

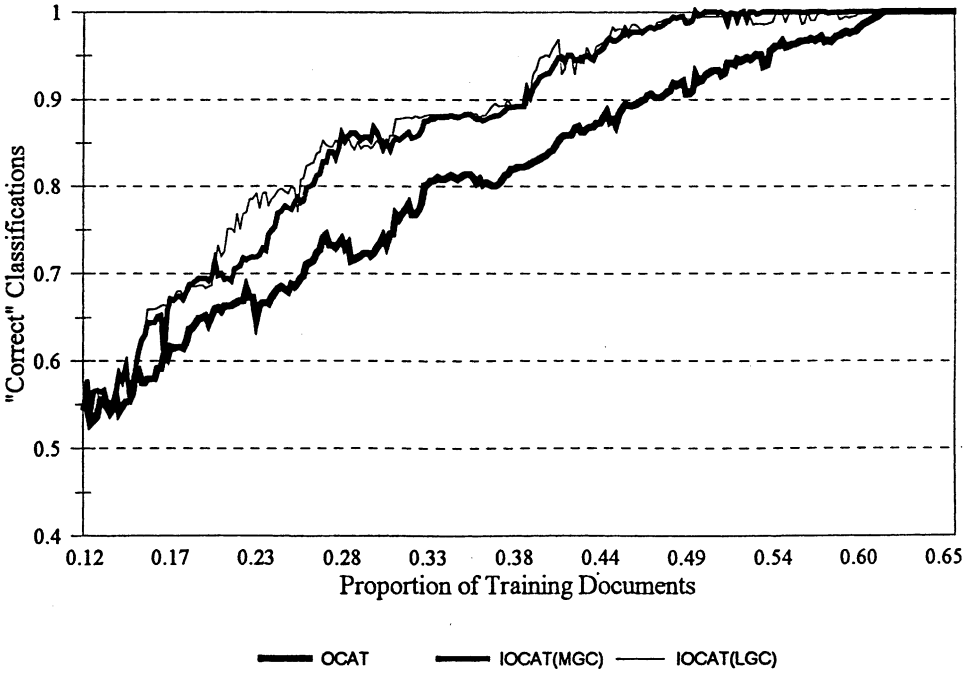

Fig. 7. Accuracy results for the class-pair (AP vs. DOE).

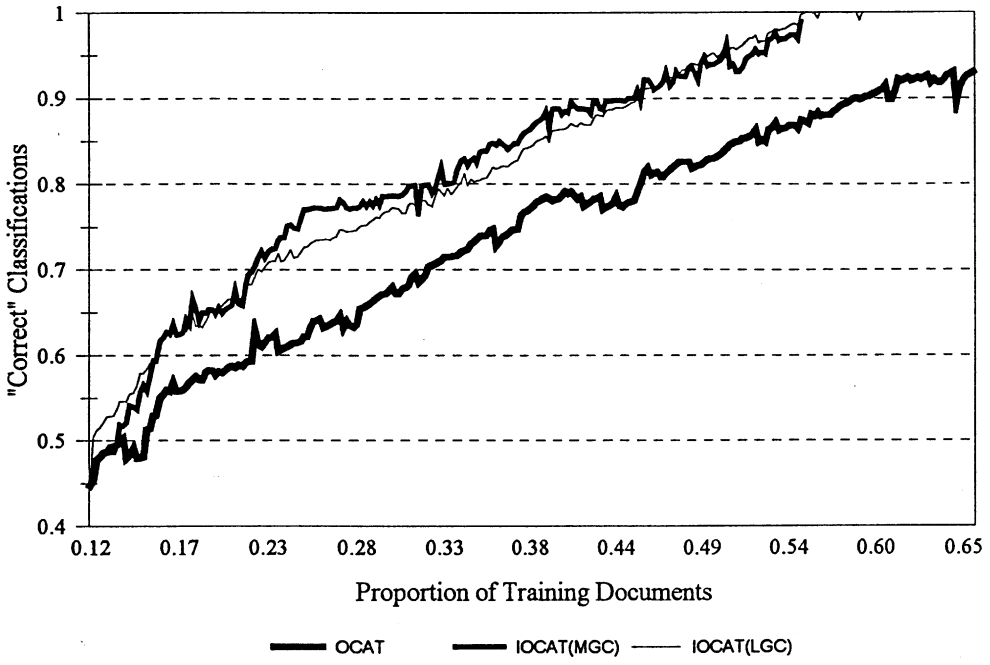

Fig. 8. Accuracy results for the class-pair (WSJ vs. ZIPFF).

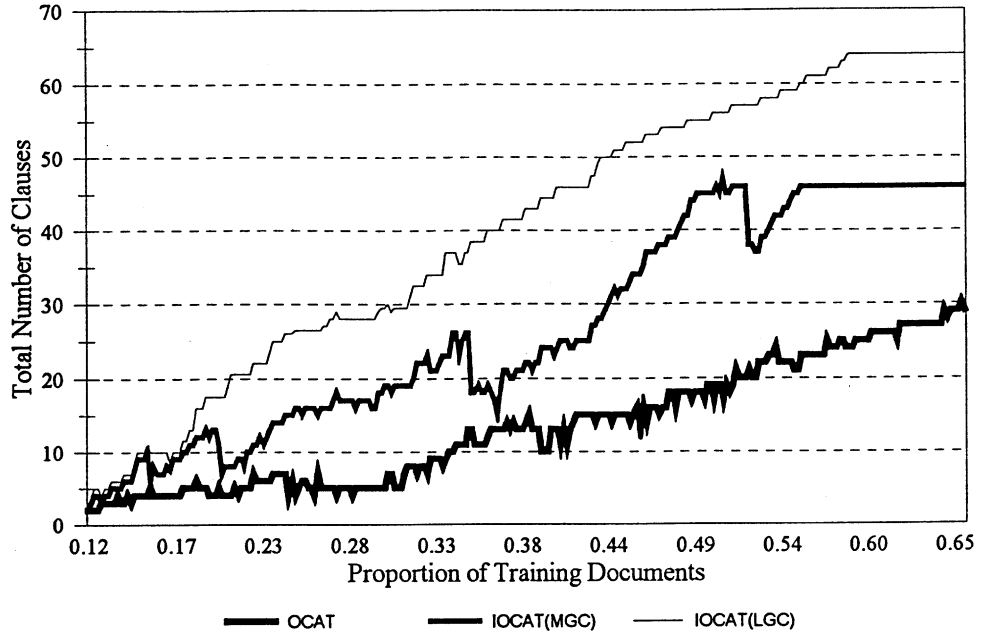

Fig. 9. Number of clauses for the class-pair (DOE vs. ZIPFF).

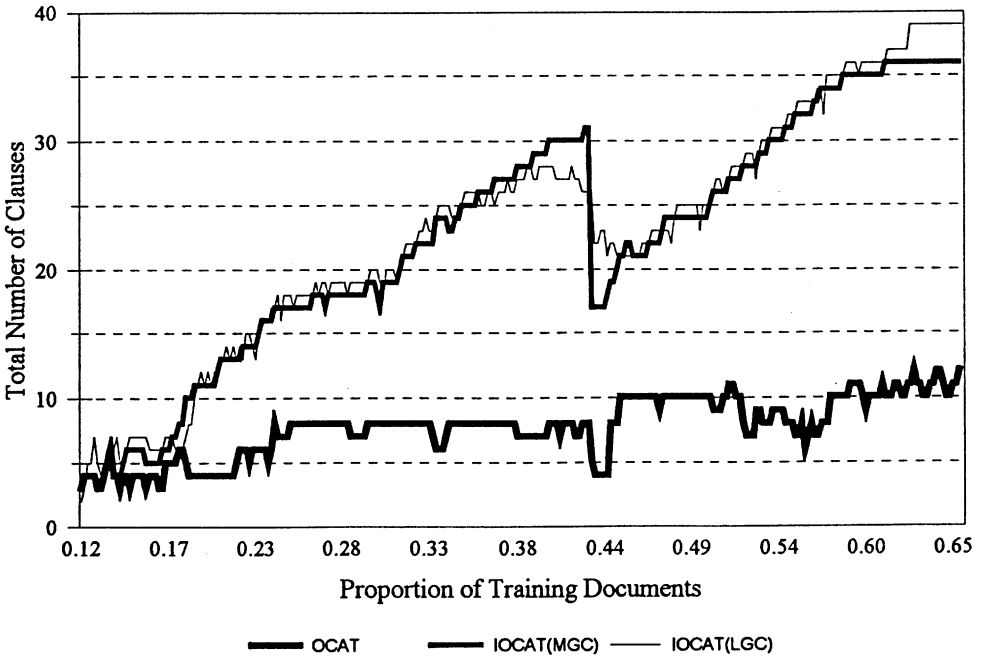

Fig. 10. Number of clauses for the class-pair (AP vs. DOE).

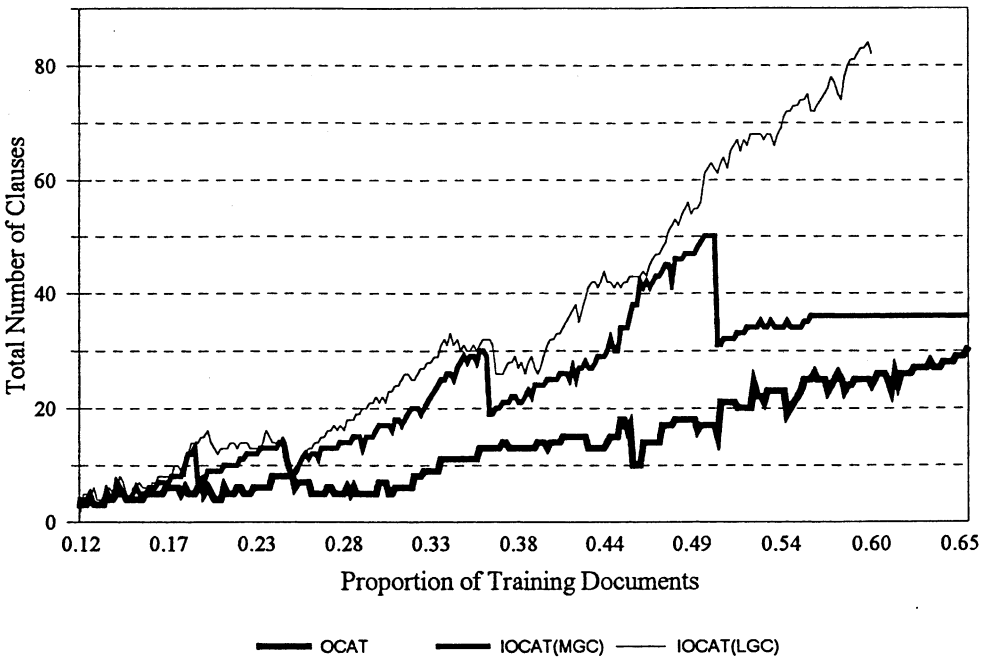

Fig. 11. Number of clauses for the class-pair (WSJ vs. ZIPFF).

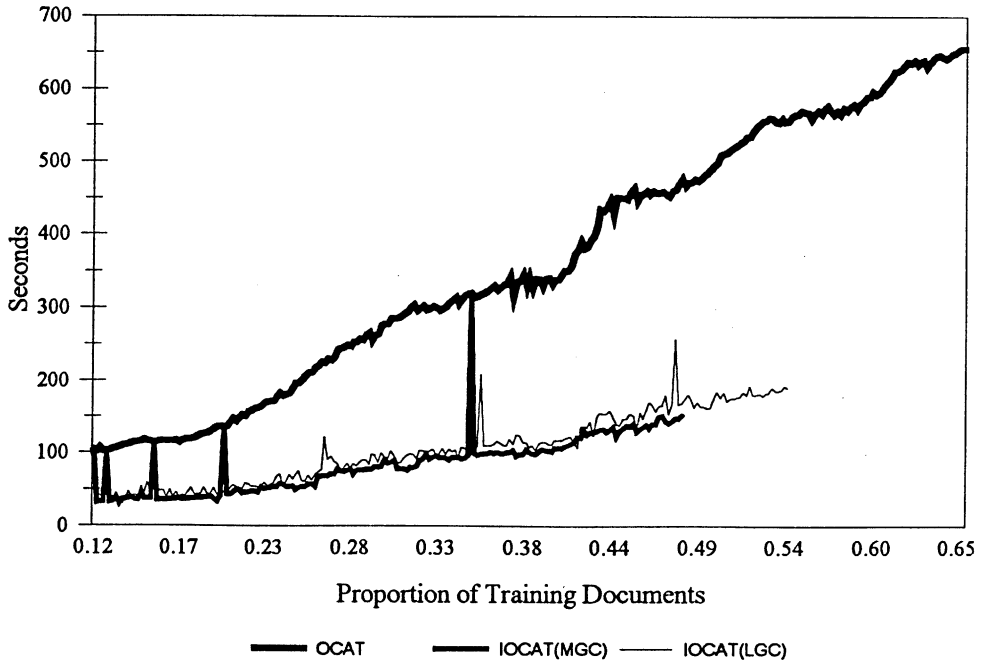

Fig. 12. Required CPU time for the class-pair (DOE vs. ZIPFF).

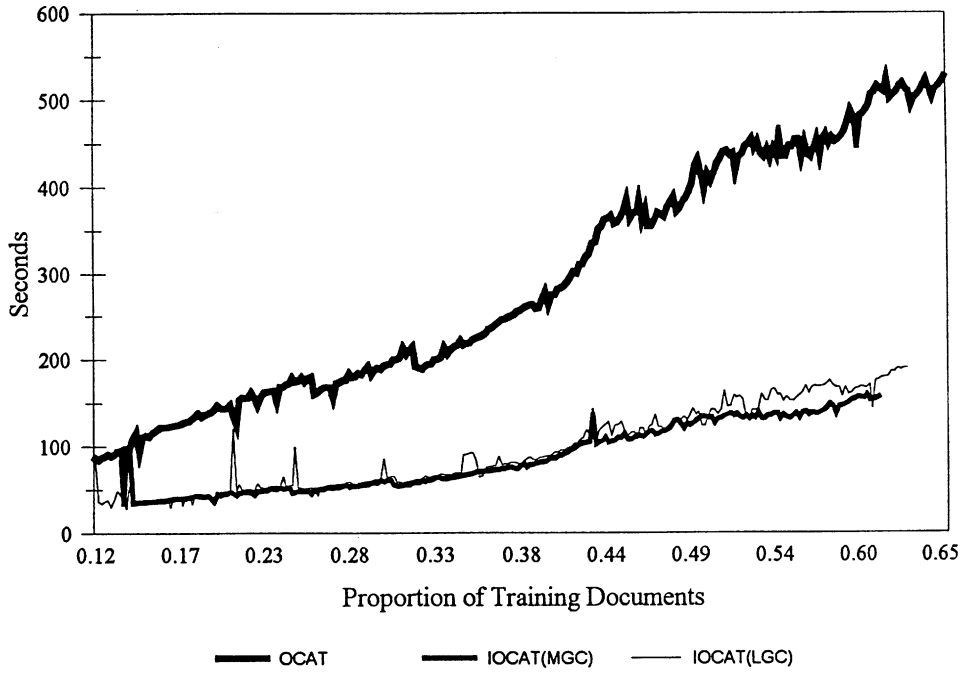

Fig. 13. Required CPU time for the class-pair (AP vs. DOE).

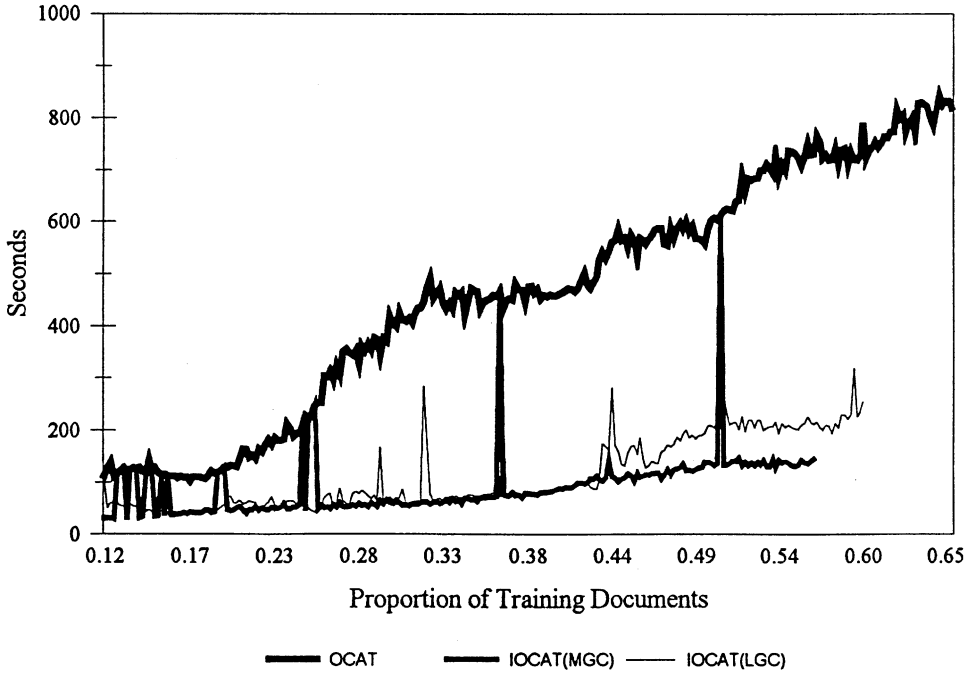

Fig. 14. Required CPU time for the class-Pair (WSJ vs. ZIPFF).

Table 2
Number of training documents to construct a rule that classified all 510 documents[a]

	Class-pair			Mean
	DOE vs. ZIPFF	AP vs. DOE	WSJ vs. ZIPFF	
OCAT	335	311	363	336
IOCAT(MGC)	244	311	288	281
IOCAT(LGC)	277	319	303	299

[a]MGC and LGC stand for the most and the least generalizing clause selection criterion, respectively.

correct classifications on the remaining unseen examples from the population of the examples in each class-pair (as shown in Table 1). Table 2 summarizes the number of training examples these learning algorithms needed to extract rules (Boolean functions) that could classify all the available examples correctly for each one of the three class-pairs from the TIPSTER collection.

The data in this table suggest that the rules constructed by the two IOCAT approaches used significantly fewer examples than the plain OCAT approach before the extracted Boolean functions could correctly classify the entire population (i.e., the training plus the unseen) of examples. Furthermore, an inspection of Figs. 6–8, indicates that the "speed" of correct classifications for IOCAT (thick and thin lines) was faster (steeper) than that for the plain OCAT approach (thickest line which is below the previous two lines). Next, in Table 3, the 0 (zero) positive signs from the comparison of the difference OCAT–IOCAT(MGC) for class-pair DOE vs. ZIPF

Table 3
Statistical comparison of the classification accuracy between OCAT and IOCAT[a]

	Number of positive signs[b]		
	DOE vs. ZIPFF	AP vs. DOE	WSJ vs. ZIPFF
OCAT–IOCAT(MGC)	0[c]	1[c]	5[c]
OCAT–IOCAT(LGC)	1[c]	2[c]	0[c]
IOCAT(MGC)–IOCAT(LGC)	92[c]	210[d]	163[d]

[a]MGC and LGC stand for the most and least generalizing clause selection criterion, respectively.
[b]All p-values were approximated using $N(np, np(1-p)^{1/2})$.
[c]Denotes a p-value close to 0. Instances $n = 250$. These different values of n were needed because all the differences yielding zero were discarded [31]. For sign test $p = 0.5$.
[d]Denotes a p-value close to 1. Instances n was, 237, 240, and 242. These different values of n were needed because all the differences yielding zero were discarded [31]. For sign test $p = 0.5$.

Table 4
Number of clauses in the rules at the end of an experiment[a]

	Class-pair			Mean
	DOE vs. ZIPFF	AP vs. DOE	WSJ vs. ZIPFF	
OCAT	23	10	25	19
IOCAT(MGC)	47	36	36	40
IOCAT(LGC)	37	60	74	57

[a]MGC and LGC stand for most and least generalizing clause selection criterion, respectively.

indicates that the ICAT(MGC) approach was always more accurate than the plain OCAT approach in all the paired observations. In contrast, the datum for IOCAT(MGC)–IOCAT(LGC) for class-pair AP vs. DOE indicates that 210 positive signs were obtained. In this case, the high number of positive signs shows that the most generalizing criterion (MGC) was a better performer than the least generalizing criterion (LGC). As it was stated in Section 4.1 this was anticipated since the MGC clause has, by definition, more potential to make an impact in the way the target Boolean function classifies examples.

The small p-value of the comparison between the two algorithms indicates that the two versions of IOCAT were much better performers than the plain OCAT approach. Second, the large p-values of the comparison of the two versions of IOCAT suggest that the two clause selection criteria performed in a similar manner.

6.2. Results on the number of clauses

The corresponding results are depicted in Figs. 9–11. These results represent the total number of clauses of the "*positive*" and "*negative*" Boolean functions. As before, some numerical highlights are summarized in Tables 4 and 5. From these results it is evident that under the plain OCAT approach the two Boolean functions had much less clauses (less than 50%) than the functions under the two ILE approaches. The same results also indicate that the IOCAT

Table 5
Statistical comparison on the number of clauses constructed by OCAT and IOCAT[a]

	Number of positive signs[b]		
	DOE vs. ZIPFF	AP vs. DOE	WSJ vs. ZIPFF
OCAT–IOCAT(MGC)	0	1	0
OCAT–IOCAT(LGC)	1	2	0
IOCAT(MGC)–IOCAT(LGC)	44[c]	0	0

[a]MGC and LGC stand for the most and least generalizing clause selection criterion, respectively.
[b]All p-values were approximated using $N(np, np(1-p)^{1/2})$.
[c]Denotes a p-value close to 0. For all instances, $n = 243$ since all differences yielding zero were discarded [31]. The p-value for the sign test was equal to 0.50.

Table 6
CPU Time (in s) required to complete an experiment[a]

	Class-pair			Mean
	DOE vs. ZIPFF	AP vs. DOE	WSJ vs. ZIPFF	
OCAT	468.532	514.584	729.430	570.848
IOCAT(MGC)	150.896	156.374	142.966	150.088
IOCAT(LGC)	178.885	168.371	215.932	187.729

[a]MGC and LGC stand for most and least generalizing clause selection criterion, respectively.

approach with the MGC selection criterion did better than the IOCAT approach when it was combined with the LGC selection criterion. As with the previous results, this can be attributed to the higher potential of the MGC clause.

An interesting phenomenon in the previous results is the occasional drop on the number of clauses under the two ILE approaches. This occurred when the ILE approach (i.e., either the IOCAT(MGC) or the IOCAT(LGC) approach) had to solve the entire function inference problem. This problem was denoted as sub-problem OCAT($E^+, E^- \cup \{e^-\}$) in Section 2. This occasional reconstruction step of the entire function alleviated the problem of generating large numbers of clauses.

6.3. Results on the CPU times

These results are depicted in Figs. 12–14, and are summarized in Tables 6 and 7. As it was anticipated, the ILE approaches (i.e., the IOCAT(MGC) and the IOCAT(LGC) approaches) required significantly less CPU time than the plain OCAT approach. This is in agreement with the discussions in Section 2 and also with similar references from the literature. For instance, [23,25] have indicated that incremental approaches always required shorter CPU times than non-incremental approaches. When the two versions of the IOCAT approach are compared, then the small p-values shown in Table 7 reveal that the MGC selection criterion always required shorter CPU times than the LGC one. This is most likely caused because the MGC can impact the behavior of the target Boolean function more significantly than the LGC one.

Table 7
Statistical comparison on the CPU time to reconstruct/modify the Boolean functions[a]

	Number of positive signs[b]		
	DOE vs. ZIPFF	AP vs. DOE	WSJ vs. ZIPFF
OCAT–IOCAT(MGC)	269	262	257
OCAT–IOCAT(LGC)	273	272	274
IOCAT(MGC)–IOCAT(LGC)	29[c]	10[c]	26[c]

[a]MGC and LGC stand for the most and least generalizing clause selection criterion, respectively.
[b]All p-values were approximated using $N(np, np(1-p)^{1/2})$.
[c]Denotes a p-value close to 0. For all instances, $n = 243$, 217, and 247 since all differences yielding zero were discarded [31]. The p-value for the sign test was equal to 0.50.

As with the results regarding the total number of the derived clauses, here too the plots have some spikes. These spikes correspond to occasions when an inference problem had to be solved on its entirety. This occurred when the set $E^+(C)$ was identical to the E^+ set in Steps 3 and 4 in Fig. 5. A similar situation also occurred in the experiments reported in [29] when a decision tree had to be rebuilt from the beginning.

7. Summary

This paper proposed an approach for inferring a Boolean function in an incremental learning environment. In such an environment, it was assumed that some training examples are available and are divided into two mutually exclusive classes. Also a "*positive*" and a "*negative*" Boolean function are available and they satisfy the requirements of the initial training data. As new examples become available, either one of the two Boolean functions may need to be modified (if it misclassifies new observations) to satisfy the requirements of the existing and also the new training data. The algorithms proposed in this paper modify a Boolean function in a localized manner, unless it is determined that the function inference problem needs to be solved on its entirety.

The proposed function modification procedures were combined with an existing algorithm for inferring a Boolean function from two classes of examples. That algorithm is the OCAT (one clause at a time) approach [1,2]. However, *any* Boolean function algorithm can be used with the proposed incremental learning approaches. An extensive empirical study was also undertaken to better assess the numerical properties of the new approaches and how they compare with the non-incremental OCAT approach. As data for the empirical study, we used binary examples that were defined by analyzing text documents from the TIPSTER collection.

The results of this investigation suggest that the proposed approaches are both effective and efficient. In these tests, the new approaches returned Boolean functions that were more accurate than the corresponding functions returned by the non-incremental OCAT approach. Furthermore, they did so in a significantly small fraction of the CPU time required by the non-incremental approach. However, as it was anticipated, the Boolean functions returned by the incremental approaches had more (slightly more than twice) clauses than under the non-incremental approach.

In summary, the results in this paper strongly suggest that if a learning task involves frequent incremental processing of large collections of examples (defined on a large set of binary attributes), then the proposed incremental learning algorithms are an effective and efficient alternative to more time consuming non-incremental approaches. It should also be stated here that these results can be extended to problems with non-binary data, since such problems can be transformed into problems defined on binary variables [37].

Acknowledgements

The authors are very thankful to the two anonymous reviewers whose thoughtful and constructive comments where pivotal in improving the quality of this paper.

Furthermore, the first two authors are very appreciative to the US Department of Energy, Office of Declassification, for providing the funds that supported this research. The first author also wishes to recognize the partial financial support from Consejo Nacional de Ciencia y Technologia-Mexico. The second author is very appreciative to the US Navy, Office of Naval Research, for the support from grants N00014-95-1-0639 and N00014-97-1-0632.

References

[1] Triantaphyllou E, Soyster AL, Kumara SRT. Generating logical expressions from positive and negative examples via a branch-and-bound approach. Computers and Operations Research 1994;21(2):185–97.
[2] Triantaphyllou E. Inference of a minimum size Boolean function from examples by using a new efficient branch-and-bound approach. Journal of Global Optimization 1994;5(1):64–94.
[3] Harman D. Overview of the second text retrieval conference (TREC-2). Information Processing and Management 1995;31(3):271–89.
[4] Voorhees E. Overview of the sixth text retrieval conference (TREC-6). Proceedings of the Sixth Text Retrieval Conference (TREC-6), MD: Gaithersburg, 1998. p. 1–27.
[5] Salton G. Automatic text processing. The transformation, analysis, and retrieval of information by computer. Reading, MA: Addison-Wesley, 1989.
[6] Valiant LG. A theory of the learnable. Communications of the ACM 1984;27(11):1134–42.
[7] Angluin, D. Learning propositional horn sentences with hints. Technical Report, YALE/DCS/RR-590, Department of Computer Science, Yale University, CN, USA, 1987.
[8] Haussler D, Warmuth M. The probably approximately correct (PAC) and other learning models. In: Meyrowitz AL, Chipman S, editors. Foundations of knowledge acquisition: Machine Learning. Norwell, MA: Kluwer Academic Publishers, 1993. p. 291–312.
[9] Holland JH, Holyoak KJ, Nisbett RE, Thagard PR. Induction: processes of inference, learning, and discovery. Cambridge, MA: MIT Press, 1986.
[10] Salton G. Automatic information organization and retrieval. New York, NY: McGraw-Hill, 1968.
[11] Cleveland D, Cleveland AD. Introduction to indexing and abstracting. Littleton, CO: Libraries Unlimited, 1983.
[12] Meadow CT. Text information retrieval systems. San Diego, CA: Academic Press, 1992.
[13] Bongard M. Pattern recognition. New York: Spartan Books, 1970.
[14] Karnaugh M. The map method for synthesis of combinatorial logic circuits. Transactions AIEE, Communications and Electronics 1965;72:593–9.
[15] Boros E, Hammer PL, Ibaraki T, Kogan A, Mayoraz E, Muchnik I. An implementation of logical analysis of data. IEEE Transactions on Knowledge and Data Engineering 2000;12(2):292–306.
[16] Brayton RK, Hachtel GD, McMullen CT, Sangiovanni-Vincentelli AL. Logic minimization algorithms for VLSI minimization. Boston, MA: Kluwer Academic Publishers, 1985.

[17] Gimpel JF. A method of producing a Boolean function having an arbitrarily prescribed prime implicant table. IEEE Transactions on Computers 1965;14:485–8.
[18] Schneeweiss W. Boolean functions with engineering applications and computer programs. Berlin, Germany: Springer, 1989.
[19] Triantaphyllou E, Soyster AL. An approach to guided learning of Boolean functions. Mathematical and Computer Modelling 1995;23(3):69–86.
[20] Triantaphyllou E, Soyster AL. A relationship between CNF and DNF systems derivable from examples. ORSA Journal on Computing 1995;7(3):283–5.
[21] Deshpande AS, Triantaphyllou E. A greedy randomized adaptive search procedure (GRASP) for inferring logical clauses from examples in polynomial time and some extensions. Mathematical and Computer Modelling 1998;27(1):75–99.
[22] Hunt E, Martin J, Stone P. Experiments in induction. New York, NY: Academic Press, 1996.
[23] Michalski RS, Larson JB. Selection of the most representative training examples and incremental generation of VL1 Hypotheses: The underlying methodology and the description of the programs ESEL and AQ11, Technical Report No. UIUCDCS-R-78-867. University of Illinois at Urbana, Department of Computer Science, Urbana, IL, USA, 1978.
[24] Michalski, RS. Knowledge repair mechanisms: evolution vs. revolution. Proceedings of the Third International Workshop on Machine Learning. Skytop, PA: 1985. p. 116–9.
[25] Reine RE, Michalski RS. Incremental learning of concept descriptions. Machine Intelligence, vol. 11. Oxford, UK: Oxford University Press, 1986.
[26] Schlimmer JC. Incremental adjustment of representations for learning. Proceedings of the Fourth International Machine Learning Workshop, Irving, CA: 1987. p. 79–90.
[27] Schlimmer, JC, Fisher D. A case study of incremental concept learning. Proceedings of the Fifth National Conference on Artificial Intelligence, Philadelphia, PA: 1986. p. 496–501.
[28] Utgoff PE. Incremental induction of decision trees. Machine Learning 1989;4:161–86.
[29] Utogff PE. Decision tree induction based on efficient tree reconstructing. Machine Learning 1998; 29:5–40.
[30] Michalski RS. Discovering classification rules using variable-valued logic system VL1. Proceedings of the Third International Joint Conference on Artificial Intelligence. Stanford, CA: 1973. p. 162–72.
[31] Banes JW. Statistical analysis for engineers and scientists. A computer-based approach. New York: McGraw-Hill, 1994.
[32] Borland. Turbo Pascal 1.5 for windows. Scotts Valley, CA: Borland International, Inc., 1991.
[33] Hsiao D, Haray F. A formal system for information retrieval from a file. Communications of the ACM 1970;13(2):67–73.
[34] Chen H. Machine learning approach to document retrieval: an overview and an experiment. Technical Report. University of Arizona, MIS Department, Tucson, AZ, USA, 1996.
[35] Chen H, Hsu R, Orwing R, Hoopes L, Numamaker JF. Automatic concept classification of text from electronic meetings. Communications of the ACM 1994;30(10):55–73.
[36] Nieto SS, Triantaphyllou E, Kraft D. A data mining approach to the classification of text documents into two disjoint Classes, Department of industrial and manufacturing systems engineering, 3128 CEBA Building, Louisiana State University, Baton Rouge, LA, 70803, submitted for publication, 2000. Web: http://www.imse.lsu.edu/vangelis/.
[37] Triantaphyllou E. The OCAT approach to knowledge discovery from examples, Working Paper, Department of Industrial and Manufacturing Systems Engineering, 3128 CEBA Building, Louisiana State University, Baton Rouge, LA, 70803, submitted for publication, 2000. Web: http://www.imse.lsu.edu/vangelis/.

Dr. Salvador Nieto Sanchez received his Ph.D. in Engineering Sciences with area of focus in Industrial Engineering from the Louisiana State University (LSU) in 1999. His research interests are in data mining, document analysis, clustering, and inventory systems. He has published refereed papers in Computers and IE and also in book chapters. Currently, he works as a top manager in the Motorola, Inc., facility in Taipei, Taiwan.

Dr. Evangelos Triantaphyllou received his Dual Ph.D. in Industrial Engineering and Operations Research from Penn State University in 1990. Currently he is an Associate Professor in the Industrial Engineering Department at

LSU and also serves as the Interim Associate Dean of Outreach and Development for the College of Engineering at LSU. His areas of research interest are in data mining and knowledge discovery from databases, multi-criteria decision making, and the interface of operations research and computer science with applications. He has extensively published in these areas. He served and is still serving in the committees of a number of international conferences and also serves in the editorial boards of some of the most prominent journals in the IE, OR, and CS areas. He has also served in many NSF and NRC panels. He and his graduate students have won many prestigious awards for excellence in research (including the 1998 Research Award in OR by IIE). He just published a new book titled "**Multi-Criteria Decision Making Methods: A Comparative Study**" (by Kluwer Academic Publishers). More details can be found in his web page at: http://www.imse.lsu.edu/vangelis/

Dr. Jianhua Chen received the Ph.D. in Computer Science from the Jieling University in China in 1988. Currently she is an Associate Professor in the Computer Science Department at LSU. Her research interests are in the areas of data mining, knowledge representation, machine learning, fuzzy logic and fuzzy systems, and intelligent databases. She has published numerous refereed papers in such journals as *Experimental and Theoretical AI*, *Logic and Computation*, *Fundamental Informatica*, *Artificial Intelligence*, *and Fuzzy Sets and Systems*. More details on Dr. Chen's activities can be found in her web page at: http://bit.csc.lsu.edu/ ~jianhua/jianhua.html

Dr. T. Warren Liao received his Ph.D. in Industrial Engineering from Lehigh University in 1990. Currently he is an Associate Professor in the Industrial Engineering Department at LSU. His areas of interest are in intelligent data analysis, data mining and knowledge discovery, fuzzy systems, clustering, and in the design of cellular manufacturing systems and he is a world-renowned authority in these areas of research and application. He has published numerous research papers in the most prestigious journals in his areas of interest and he has served as a Guest Editor for a number of prominent journals. Dr. Liao has received significant funding from state and federal agencies to conduct research in these areas. More details on his academic and research activities can be found in his web page at: http://www.imse.lsu.edu/liao/