DIAGARA: AN INCREMENTAL ALGORITHM FOR INFERRING IMPLICATIVE RULES FROM EXAMPLES

Xenia Naidenova

Abstract: *An approach is proposed for inferring implicative logical rules from examples. The concept of a good diagnostic test for a given set of positive examples lies in the basis of this approach. The process of inferring good diagnostic tests is considered as a process of inductive common sense reasoning. The incremental approach to learning algorithms is implemented in an algorithm DIAGaRa for inferring implicative rules from examples.*

Keywords: *Incremental and non-incremental learning, learning from examples, machine learning, common sense reasoning, inductive inference, good diagnostic test, lattice theory.*

ACM Classification Keywords: *I.2.6 Artificial Intelligence: Learning; K.2.3. Concept Learning*

Introduction

Our approach to machine learning problems is based on the concept of a good diagnostic (classification) test. This concept has been advanced firstly in the framework of inferring functional and implicative dependencies from relations [Naidenova and Polegaeva, 1986]. But later the fact has been revealed that the task of inferring all good diagnostic tests for a given set of positive and negative examples can be formulated as the search of the best approximation of a given classification on a given set of examples and that it is this task that all well known machine learning problems can be reduced to [Naidenova, 1996].

We have chosen the lattice theory as a model for inferring good diagnostic tests from examples from the very beginning of our work in this direction. We believe that it is the lattice theory that must be the mathematical theory of common sense reasoning. One can come to this conclusion by analyzing both the fundamental work in the psychological theory of intelligence [Piaget, 1959], and the experience of modeling thinking processes in the framework of artificial intelligence. The process of objects' classification has been considered in [Shreider, 1974] as an algebraic idempotent semi group with the unit element. An algebraic model of classification and pattern recognition based on the lattice theory has been advanced in [Boldyrev, 1974]. A lot of experience has been obtained on the application of algebraic lattices in machine learning: the works of Finn and his disciples [Finn, 1984], [Kuznetsov, 1993], the model of conceptual knowledge of Wille [1992], the works of the French group [Ganascia, 1989]. The following works are devoted to the application of algebraic lattices for extracting classifications, functional dependencies and implications from data: [Demetrovics and Vu, 1993], [Mannila and Räihä, 1992], [Mannila and Räihä, 1994], [Huntala, et al., 1999], [Cosmadakis, et al., 1986], [Naidenova and Polegaeva, 1986], [Megretskaya, 1989], [Naidenova, et al., 1995a], [Naidenova, et al., 1995b], and [Naidenova, 1992].

An advantage of the algebraic lattices approach is based on the fact that an algebraic lattice can be defined both as an algebraic structure that is declarative and as a system of dual operations with the use of which the elements of this lattice can be generated. This approach allows us to investigate the processes of inferring good classification tests as inductive reasoning processes. In the following part of this chapter, we shall describe our decomposition of the inductive inferring process into subtasks and operations that conform to the operations and subtasks of the natural human reasoning process.

This paper is organized as follows. The concept of a good diagnostic test is introduced and the problem of inferring all good diagnostic tests for a given classification on a given set of examples is formulated. The next section contains the description of a mathematical model underlying algorithms of learning reasoning. We propose a decomposition of learning algorithms into operations and subtasks that are in accordance with human reasoning operations. In the second part of this paper, the concepts of an essential value and an essential example are also introduced and an incremental learning algorithm DIAGaRa is described. The paper ends with a brief summary section.

The Concept of a Good Classification Test

Our approach for inferring implicative rules from examples is based on the concept of a good classification test. A good classification test can be understood as an approximation of a given classification on a given set of examples [Naidenova, 1996]. On the other hand, the process of inferring good tests realizes one of the known canons of induction formulated by J. S. Mill, namely, the joint method of similarity-distinction [Mill, 1900].

A good diagnostic test for a given set of examples is defined as follows. Let R be a table of examples and S be the set of indices of examples belonging to R. Let $R(k)$ and $S(k)$ be the set of examples and the set of indices of examples from a given class k, respectively.

Denote by $FM = R/R(k)$ the examples of the classes different from class k. Let U be the set of attributes and T be the set of attributes values (values, for short) each of which appears at least in one of the examples of R. Let n be the number of examples of R. We denote the domain of values for an attribute Atr by $dom(Atr)$, where $Atr \in U$.

By $s(a)$, $a \in T$, we denote the subset $\{i \in S$: 'a' appears in t_i, $t_i \in R\}$, where $S = \{1, 2, .., n\}$.

Following [Cosmadakis, et al., 1986], we call $s(a)$ the interpretation of $a \in T$ in R. It is possible to say that $s(a)$ is the set of indices of all the examples in R which are covered by the value a.

Since for all a, $b \in dom(Atr)$, $a \neq b$ implies that the intersection $s(a) \cap s(b)$ is empty, the interpretation of any attribute in R is a partition of S into a family of mutually disjoint blocks. By $P(Atr)$, we denote the partition of S induced by the values of an attribute Atr. The definition of $s(a)$ can be extended to the definition of $s(t)$ for any collection t of values as follows: for t, $t \subseteq T$, if $t = a_1 a_2 \dots a_m$, then $s(t) = s(a_1) \cap s(a_2) \cap \dots \cap s(a_m)$.

Definition 1. A collection $t \subseteq T$ $(s(t) \neq \varnothing)$ of values, is a diagnostic test for the set $R(k)$ of examples if and only if the following condition is satisfied: $t \not\subset t^*$, $\forall$ t^*, $t^* \in FM$ (the equivalent condition is $s(t) \subseteq S(k)$).

To say that a collection t of values is a diagnostic test for the set $R(k)$ is equivalent to say that it does not cover any example belonging to the classes different from k. At the same time, the condition $s(t) \subseteq S(k)$ implies that the following implicative dependency is true: 'if t, then k.

It is clear that the set of all diagnostic tests for a given set $R(k)$ of examples (call it '$DT(k)$') is the set of all the collections t of values for which the condition $s(t) \subseteq S(k)$ is true. For any pair of diagnostic tests t_i, t_j from $DT(k)$, only one of the following relations is true: $s(t_i) \subseteq s(t_j)$, $s(t_i) \supseteq s(t_j)$, $s(t_i) \approx s(t_j)$, where the last relation means that $s(t_i)$ and $s(t_j)$ are incomparable, i.e. $s(t_i) \not\subset s(t_j)$ and $s(t_j) \not\subset s(t_i)$. This consideration leads to the concept of a good diagnostic test.

Definition 2. A collection $t \subseteq T$ $(s(t) \neq \varnothing)$ of values is a good test for the set $R(k)$ of examples if and only if the following condition is satisfied: $s(t) \subseteq S(k)$ and simultaneously the condition $s(t) \subset s(t^*) \subseteq S(k)$ is not satisfied for any t^*, $t^* \subseteq T$, such that $t^* \neq t$.

Good diagnostic tests possess the greatest generalization power and give a possibility to obtain the smallest number of implicative rules for describing examples of a given class k.

The Characterization of Classification Tests

Any collection of values can be irredundant, redundant or maximally redundant.

Definition 3. A collection t of values is irredundant if the following condition is satisfied: $(\forall v)$, $(v \in t)$, $s(t) \subset s(t/v)$.

If a collection t of values is a good test for $R(k)$ and, simultaneously, it is an irredundant collection of values, then any proper subset of t is not a test for $R(k)$.

Definition 4. Let $X \rightarrow v$ be an implicative dependency which is satisfied in R between a collection $X \subseteq T$ of values and the value v, $v \in T$. Suppose that a collection $t \subseteq T$ of values contains X. Then the collection t is said to be redundant if it contains also the value v.

If t contains the left and the right sides of some implicative dependency $X \rightarrow v$, then the following condition is satisfied: $s(t) = s(t/v)$. In other words, a redundant collection t and the collection t/v of values cover the same set of examples.

If a good test for $R(k)$ is a redundant collection of values, then some values can be deleted from it and thus obtain an equivalent good test with a smaller number of values.

Definition 5. A collection $t \subseteq T$ of values is maximally redundant if for any implicative dependency $X \rightarrow v$, which is satisfied in *R*, the fact that *t* contains *X* implies that *t* also contains *v*.

If *t* is a maximally redundant collection of values, then for any value $v \notin t$, $v \in T$ the following condition is satisfied: $s(t) \supset s(t \cup v)$. In other words, a maximally redundant collection *t* of values covers the number of examples greater than the collection $(t \cup v)$ of values.

Any example *t* in *R* is a maximally redundant collection of values because for any value $v \notin t$, $v \in T$ $s(t \cup v)$ is equal to $\varnothing$.

If a diagnostic test for a given set *R*(*k*) of examples is a good one and it is a maximally redundant collection of values, then by adding to it any value not belonging to it we get a collection of values which is not a good test for *R*(*k*).

Table - 1. Example 1 of Data Classification. (This example is adopted from [Ganascia, 1989]).

Index of Example	Height	Color of Hair	Color of Eyes	Class
1	Short	Blond	Blue	1
2	Short	Brown	Blue	2
3	Tall	Brown	Embrown	2
4	Tall	Blond	Embrown	2
5	Tall	Brown	Blue	2
6	Short	Blond	Embrown	2
7	Tall	Red	Blue	1
8	Tall	Blond	Blue	1

For example, in Table 1 the collection '*Blond Blue*' is a good irredundant test for class 1 and simultaneously it is maximally redundant collection of values. The collection '*Blond Embrown*' is a test for class 2 but it is not good test and simultaneously it is maximally redundant collection of values.

The collection '*Embrown*' is a good irredundant test for class 2. The collection '*Red*' is a good irredundant test and the collection '*Tall Red Blue*' is a maximally redundant and good test for class 1.

It is clear that the best tests for pattern recognition problems must be good irredundant tests. These tests allow construction of the shortest implicative rules with the highest degree of generalization.

An Approach for Constructing Good Irredundant Tests

Let *R*, *S*, *S*(+), *T*, *s*(*t*), $t \subseteq T$, $s \subseteq S$ be as defined earlier. We give the following propositions the proof of which can be found in [Naidenova, 1999].

PROPOSITION 1.

The intersection of maximally redundant collections of values is a maximally redundant collection.

PROPOSITION 2.

Every collection of values is contained in one and only one maximally redundant collection with the same interpretation.

PROPOSITION 3.

A good maximal redundant test for R(*k*) *either belongs to the set R*(*k*) *or it is equal to the intersection of q examples from R*(*k*) *for some q, $2 \le q \le nt$, where nt is the number of examples in R*(*k*).

One of the possible ways for searching for good irredundant tests for a given class of examples is the following: first, find all good maximally redundant tests; second, for each good maximally redundant test, find all good irredundant tests contained in it. This is a convenient strategy as each good irredundant test belongs to one and only one good maximally redundant test with the same interpretation.

It should be more convenient in the following considerations to denote the set *R*(*k*) as *R*(+) (the set of positive examples) and the set *R*/*R*(*k*) as *R*(-) (the set of negative examples). We will also denote the set *S*(*k*) as *S*(+).

The following Algorithm 1 solves the task of inferring all good maximally redundant tests for a given set of positive examples. The idea of this algorithm has been advanced in [Naidenova and Polegaeva, 1991].

By $s_q = (i_1, i_2, \ldots, i_q)$, we denote a subset of S, containing q indices from S. Let S(test-q) be the set of elements $s = \{i_1, i_2, \ldots, i_q\}$, $q = 1, 2, \ldots, nt$, satisfying the condition that $t(s)$ is a test for $R(+)$. Here nt denotes the number of positive examples.

We will use an inductive rule for constructing $\{i_1, i_2, \ldots, i_{q+1}\}$ from $\{i_1, i_2, \ldots, i_q\}$, $q = 1, 2, \ldots, nt-1$. This rule relies on the following consideration: if the set $\{i_1, i_2, \ldots, i_{q+1}\}$ corresponds to a test for $R(+)$, then all its proper subsets must correspond to tests too and, consequently, they must be in S(test-q). Thus the set $\{i_1, i_2, \ldots, i_{q+1}\}$ can be constructed if and only if S(test-q) contains all its proper subsets. Having constructed the set $s_{q+1} = \{i_1, i_2, \ldots, i_{q+1}\}$, we have to determine whether it corresponds to the test or not. If $t(s_{q+1})$ is not a test, then s_{q+1} is deleted, otherwise s_{q+1} is inserted in S(test-$(q+1)$). The algorithm is over when it is impossible to construct any element for S(test-$(q+1)$).

We use in Algorithm 1 the function to_be_test(t): if $s(t) \cap S(+) = s(t)$ ($s(t) \subseteq S(+)$) then *true* else *false*.

We introduce the mapping $t(s)$ = {intersection of all t_i: $t_i \subseteq T$, $i \in s$}.

Algorithm 1. Inferring all Good Maximally Redundant Tests (GMRTs) for a set $R(+)$ of positive examples.

```
1. Input: $q = 1$, $R$, $S$, $R(+)$, $S(+) = \{1, 2, \ldots, nt\}$, $S$(test-$q$) = $\{\{1\}, \{2\}, \ldots, \{nt\}\}$.
Output: the set $TGOOD$ of all GMRTs for $R(+)$.
2. $S_q$ ::= $S$(test-$q$);
3. While $||S_q|| \geq q + 1$ do
3.1 Generating $S(q + 1) = \{s = \{i_1, \ldots, i_{(q+1)}\}: (\forall j)\ (1 \leq j \leq q + 1)\ (i_1, \ldots, i_{(j-1)}, i_{(j+1)}, \ldots, i_{(q+1)}\} \in S_q\}$;
3.2 Generating $S$(test-$(q + 1)$) = $\{s = \{i_1, \ldots, i_{(q+1)}\}: (s \in S(q + 1))$ & (to_be_test($t(s)$) = $true$)$\}$;
3.3 $S$(test-$q$) ::= $\{s = \{i_1, \ldots, i_q\}: (s \in S$(test-$q$)) & $((\forall s')(s' \in S$(test-$(q + 1)$)) $s \not\subset s')\}$;
3.4. $q ::= q + 1$;
3.5. $max ::= q$;
end while
4. $TGOOD ::= \varnothing$;
5. While $q \leq max$ do $TGOOD ::= TGOOD \cup \{t(s): s = \{i_1, \ldots, i_s\} \in S$(test-$q$) $\}$;
5.1 $q ::= q + 1$;
end while
end
```

An illustration of inferring GMRTs for the examples of class 2 (see, please, Table 1) is given in Table 2.

The set S_q, $q = 2$ consists of 10 elements {{2,3}, {2,4}, {2,5}, {2,6}, {3,4}, {3,5}, {3,6}, {4,5}, {4,6}, {5,6}}. But $t(\{2,4\})$, $t(\{2,6\})$, $t(\{4,5\})$, and $t(\{5,6\})$ are not tests for class2, hence we can construct only two elements of the next level for $q = 3$: $S_3 = S$(test-3) = {{2,3,5}, {3,4,6}}.

As a result, the tests obtained correspond to the following implicative rules: "if COLOR of HAIR = *Brown*, then Class = 2" and "if COLOR of EYES = *Embrown*, then Class = 2".

Algorithm 1 is also used for inferring all good irredundant tests (GIRTs) contained in a good maximally redundant test.

Now let $t = \{a_1, a_2, \ldots, a_m\} \subseteq T$ be a collection of values that is a GMRT for $R(+)$.

We will use a rule of inductive transition from an element $t_q = (a_1, a_2, \ldots, a_q)$ to another element $t_{q+1} = (a_1, a_2, \ldots, a_{q+1})$, $t_q, t_{q+1} \subseteq T$. But now we are interested in obtaining irredundant collections of values. If $t_{q+1} = (a_1, a_2, \ldots, a_{q+1})$ is irredundant, then all its proper subsets must be irredundant too.

Table - 2. Example of inferring logical rules for Class 2 (Table 1) with the use of Algorithm 1.

S(test-1)	$t(s)$, $s \in S$(test-1)	S(test-2)	$t(s)$, $s \in S$(test-2)	S(test-3)	$t(s)$, $s \in S$(test-3)
{2}	'*Short Brown Blue*'	{2,3}	'*Brown*'	{2,3,5}	'*Brown*'
{3}	'*Tall Brown Embrown*'	{2,5}	'*Brown Blue*'		

{4}	'*Tall Blond Embrown*'	{3,4}	'*Tall Embrown*'	{3,4,6}	'*Embrown*'
{5}	'*Tall Brown Blue*'	{3,5}	'*Tall Brown*'		
{6}	'*Short Blond Embrown*'	{3,6}	'*Embrown*'		
		{4,6}	'*Blond Embrown*'		

Having constructed the set $t_{q+1} = (a_1, a_2, \ldots, a_{q+1})$, we have to determine whether it is an irredundant collection of values or not. If t_{q+1} is redundant, then it is deleted, if t_{q+1} is a test, then t_{q+1} is inserted in the set *TGOOD* of all good irredundant tests contained in *t*. If t_{q+1} is irredundant but not a test, then it is a candidate for extension.

The following Algorithm 2 solves the task of inferring all GIRTs contained in a maximally redundant test for a given set of positive examples.

We use in Algorithm 2 the function to_be_irredundant(t)::= if for $(\forall\ a_i)$ $(a_i \in t)$ $s(t) \neq s(t/\ a_i)$ then *true* else *false*.

Algorithm 2. Inferring all GIRTs contained in a given GMRT for $R(+)$.

Input: $q = 1$, R, S, $R(+)$, $t = \{a_1, a_2, \ldots, a_m\}$ – a collection of values – a GMRT, F(irredundant – q) = $\{\{a_1\}, \{a_2\}, \ldots, \{a_m\}\}$ – the family of irredundant subsets of values with q equal to 1.

Output: the set *TGOOD* of all the GIRTs for $R(+)$ contained in t.

```
1. $F_q$::= $F$(irredundant – $q$ );
1.1 Generating $F$(test-$q$ ) =$\{t = \{a_{i1}, \ldots, a_{iq}\}$: $(t \in F_q)$ & (to_be_test($t$) = true)$\}$;
1.2 $F_q$ ::= $F_q \setminus F$(test-$q$) ;
2. While $\mid \mid F_q \mid \mid \geq q + 1$ do
2.1. Generating $F(q + 1)$ =
= $\{t = \{a_{i1}, \ldots, a_{i(q+1)}\}$: $(\forall\ j)$ $(1 \leq j \leq q + 1)$ $(a_{i1}, \ldots, a_{i(j-1)}, a_{i(j+1)}, \ldots, a_{i(q+1)}\} \in F_q\}$;
2.2. Generating $F$(irredundant – $(q+1)$) :
$F$(irredundant – $(q+1)$) ::= $\{t \in F(q + 1)$: to_be_irredundant($t$) = true $\}$;
2.3. $q$ ::= $q + 1$;
2.4. max ::= $q$;
end while
3. TGOOD ::= $\varnothing$;
4.While $q \leq$ max do
4.1. TGOOD ::= TGOOD $\cup$ $F$(test-$q$);
4.2. $q$::= $q + 1$;
end while
end
```

The Duality of Good Diagnostic Tests

In Algorithms 1 and 2, we used (without explicit definition) correspondences of Galois G on $S \times T$ and two relations $S \rightarrow T$, $T \rightarrow S$ [Ore, 1944], [Riguet, 1948]. Let $s \subseteq S$, $t \subseteq T$. We define the relations as follows:

$S \rightarrow T$: $t(s) = \{$intersection of all t_i: $t_i \subseteq T$, $i \in s\}$ and $T \rightarrow S$: $s(t) = \{i$: $i \in S$, $t \subseteq t_i\}$.

Extending s by an index j^* of some new example leads to receiving a more general feature of examples:

$(s \cup j^*) \supseteq s$ implies $t(s \cup j^*) \subseteq t(s)$.

Extending t by a new value 'a' leads to decreasing the number of examples possessing the general feature 'ta' in comparison with the number of examples possessing the general feature 't':

$(t \cup a) \supseteq t$ implies $s(t \cup a) \subseteq s(t)$.

We introduce the following generalization operations (functions):

generalization_of(t) = $t' = t(s(t))$; generalization_of(s) = $s' = s(t(s))$.

As a result of the generalization of s, the sequence of operations $s \rightarrow t(s) \rightarrow s(t(s))$ gives that $s(t(s)) \supseteq s$. This generalization operation gives all the examples possessing the feature $t(s)$.

As a result of the generalization of t, the sequence of operations $t \to s(t) \to t(s(t))$ gives that $t(s(t)) \supseteq t$. This generalization operation gives the maximal general feature for examples the indices of which are in $s(t)$.

These generalization operations are not artificially constructed operations. One can perform mentally a lot of such operations during a short period of time. We give some examples of these operations. Suppose that somebody has seen two films (s) with the participation of Gerard Depardieu ($t(s)$). After that, he tries to know all the films with his participation ($s(t(s))$). One can know that Gerard Depardieu acts with Pierre Richard (t) in several films ($s(t)$). After that, he can discover that these films are the films of the same producer Francis Veber $t(s(t))$.

Namely, these generalization operations will be used in the algorithm DIAGaRa.

The Definition of Good Diagnostic Tests as Dual Objects

We implicitly used two generalization operations in all the considerations of diagnostic tests. Now we define a diagnostic test as a dual object, i.e. as a pair (SL, TA), $SL \subseteq S$, $TA \subseteq T$, $SL = s(TA)$ and $TA = t(SL)$.

The task of inferring tests is a dual task. It must be formulated both on the set of all subsets of S, and on the set of all subsets of T.

Definition 6. Let $PM = \{s_1, s_2, \ldots, s_m\}$ be a family of subsets of some set M. Then PM is a Sperner system [Sperner, 1928] if the following condition is satisfied: $s_i \not\subset s_j$ and $s_j \not\subset s_i$, $\forall(i,j)$, $i \neq j$, $i, j = 1, \ldots, m$.

Definition 7. To find all *Good Maximally Redundant Tests* (GMRTs) for a given class $R(k)$ of examples means to construct a family PS of subsets $s_1, s_2, \ldots, s_j, \ldots, s_{np}$ of the set $S(k)$ such that:

1) PS is a Sperner system;

2) Each s_j is a maximal set in the sense that adding to it the index i of example t_i such that $i \notin s_j$, $i \in S$ implies $s(t(s_j \cup i)) \not\subset S(k)$. Putting it in another way, $t(s_j \cup i)$ is not a test for the class k, so there exists such example t^*, $t^* \in R(\text{-})$ that $t(s_j \cup i) \subseteq t^*$.

The set of all GMRTs is determined as follows: $\{t$: $t(s_j)$, $s_j \in PS$, $\forall j$, $j = 1, \ldots, np\}$.

Definition 8. To find all *Good Irredundant Tests* (GIRTs) for a given class $R(k)$ of examples means to find a family PRT of subsets $t_1, t_2, \ldots, t_j, \ldots, t_{nq}$ of the set T such that:

1) $t_j \not\subset t$ $\forall j$, $j = 1, \ldots, nq$, $\forall t$, $t \in R/\ R(k)$ and, simultaneously, $\forall t_j$, $j = 1, \ldots, nq$, $s(t_j) \neq \varnothing$ there does not exist such a collection $s^* \neq s(t_j)$, $s^* \subseteq S$ of indices for which the following condition is satisfied $s(t_j) \subset s^* \subseteq S(k)$;

2) PRT is a Sperner system;

3) Each t_j – a minimal set in the sense that removing from it any value $a \in t_j$ implies $s(t_j$ without $a) \not\subset S(k)$.

Decomposition of Good Classification Tests Inferring into Subtasks

The Algorithms 1 and 2 find all the GMRTs and GIRTs for a given set of positive examples but the number of tests can be exponentially large. In this case, these algorithms will be not realistic. Now we consider some decompositions of the problem that provide the possibility to restrict the domain of searching, to predict, in some degree, the number of tests, and to choose tests with the use of essential values and/or examples. This decomposition gives an approach to constructing incremental algorithms of inferring all good classification tests for a given set of examples.

We consider two kinds of subtasks (please, see also [Naidenova, 2001]:

for a given set of positive examples

1) Given a positive example t, find all GMRTs contained in t;

2) Given a non-empty collection of values X (maybe only one value) such that it is not a test, find all GMRTs containing X.

Each example contains only some subset of values from T, hence each subtask of the first kind is simpler than the initial one. Each subset X of T appears only in a part of all examples; hence each subtask of the second kind is simpler than the initial one.

Forming the Subtasks

The subtask of the first kind. We introduce the concept of an example's projection proj(R)[t] of a given positive example t on a given set R(+) of positive examples. The proj(R)[t] is the set Z = {z: (z is non-empty intersection of t and t') & ($t' \in R$(+)) & (z is a test for a given class of positive examples)}.

If the proj(R)[t] is not empty and contains more than one element, then it is a subtask for inferring all GMRTs that are in t. If the projection contains one and only one element equal to t, then t is a GMRT.

To make the operation of forming a projection perfectly clear we construct the projection of t_2 = '*Short Brown Blue*' on the examples of the second class (Table 1). This projection includes t_2 and the intersections of t_2 with the other positive examples of the second class, i.e. with the examples t_3, t_4, t_5, t_6 (Table 3).

Table - *3*. The Intersections of Example t_2 with the Examples of Class 2.

Index of Example	*Height*	*Color of Hair*	*Color of Eyes*	*Test?*
2	*Short*	*Brown*	*Blue*	*Yes*
3		*Brown*		*Yes*
4				*No*
5		*Brown*	*Blue*	*Yes*
6	*Short*			*No*

In order to check whether an element of the projection is a test or not we use the function to_be_test(t) in the following form: to_be_test(t) = if $s(t) \subseteq S(+)$ then *true* else *false*, where S(+) is the set of indices of positive examples, $s(t)$ is the set of indices of all positive and negative examples containing t. If S(-) is the set of indices of negative examples, then $S = S(+) \cup S(-)$ and $s(t) = \{i: t \subseteq t_i, i \in S\}$.

Table - 4. The Projection of the Example t_2 on the Examples of Class 2.

Index of Example	Height	Color of Hair	Color of Eyes	Test?
2	Short	Brown	*Blue*	Yes
3		Brown		Yes
5		Brown	Blue	Yes

The intersection $t_2 \cap t_4$ is the empty set. Hence, the row of the projection with the number 4 is empty. The intersection $t_2 \cap t_6$ is not a test for Class 2 because s(*Short*) = {1,2,6} $\not\subset S$(+), where S(+) is equal to {2,3,4,5,6}.

Finally, we have the projection of t_2 on the examples of the second class in Table 4.

The subtask turns out to be very simple because the intersection of all the rows of the projection is a test for the second class: t({2,3,5}) = '*Brown*', s(*Brown*) = {2,3,5} $\subseteq S$(+).

The subtask of the second kind. We introduce the concept of an attributive projection proj(R)[a] of a given value 'a'on a given set R(+) of positive examples.

The projection proj(R)[a] = {t: ($t \in R$(+)) & ('a' appears in t)}. Another way to define this projection is: proj(R)[a] = $\{t_i: i \in (s(a) \cap S(+))\}$. If the attributive projection is not empty and contains more than one element, then it is a subtask of inferring all GMRTs containing a given value 'a'. If 'a' appears in one and only one example, then 'a' does not belong to any GMRT different from this example.

Forming the projection of 'a' makes sense if 'a' is not a test and the intersection of all positive examples in which 'a' appears is not a test too, i.e. $s(a) \not\subset S(+)$ and $t' = t(s(a) \cap S(+))$ is also not a test for a given set of positive examples.

Denote the set $\{s(a) \cap S(+)\}$ by splus(a). In Table 1, we have:

S(+) = {2,3,4,5,6}, *splus*(*Short*) $\rightarrow$ {2,6}, *splus*(*Brown*) $\rightarrow$ {2,3,5}, *splus*(*Blue*) $\rightarrow$ {2,5}, *splus*(*Tall*) $\rightarrow$ {3,4,5}, *splus*(*Embrown*) $\rightarrow$ *{3,4,6},* and *splus*(*Blond*) $\rightarrow$ *{4,6}.*

For the value '*Brown*' we have: s(*Brown*) = {2,3,5} and s(*Brown*) = *splus*(*Brown*), i.e. s(*Brown*) $\subseteq S$(+).

*Analogously for the value '*Embrown*' we have:* s*(*Embrown*) = {3,4,6} and* s*(*Embrown*) =* splus*(*Embrown*), i.e.* s*(*Embrown*)* $\subseteq$ S*(+).*

Table - *5*. The Result of Reducing the Projection after Deleting the Values '*Brown*' and '*Embrown*'

Index of Example	Height	Color of Hair	Color of Eyes	Test?
2	Short		Blue	No
3	Tall			No
4	Tall	Blond		No
5	Tall		Blue	No
6	Short	Blond		No

These values are irredundant and simultaneously maximally redundant tests because *t*({2,3,5}) = '*Brown*' and *t*({3,4,6}) = '*Embrown*'. It is clear that these values cannot belong to any test different from them. We delete '*Brown*' and '*Embrown*' from further consideration with the following result as shown in Table 5.

Now none of the remaining rows of the second class is a test because *s*(*Short*, *Blue*) = {1,2}, *s*(*Tall*) = {3,4,5,7,8}, *s*(*Tall*, *Blond*) = {4,8}, *s*(*Tall*, *Blue*) ={5,7,8}, *s*(*Short*, *Blond*) = {1,6} $\not\subset$ *S*(+). The values '*Brown*' and '*Embrown*' exhaust the set of the GMRTs for this class of positive examples.

Reducing the Subtasks

The following theorem gives the foundation for reducing projections both of the first and the second kind. The proof of this theorem can be found in [Naidenova et al., 1995b].

THEOREM 1.

Let A be a value from T, X be a maximally redundant test for a given set R(+) *of positive examples and s*(*A*) $\subseteq$ *s*(*X*). *Then A does not belong to any maximally redundant good test for R*(+) *different from X.*

To illustrate the way of reducing projections, we consider another partition of the rows of Table 1 (see, please Part 1 of this paper) into the sets of positive and negative examples as shown in Table 6.

Let *S*(+) be equal to {4,5,6,7,8}. The value '*Red*' is a test for positive examples because *s*(*Red*) = *splus*(*Red*) = {7}. Delete '*Red*' from the projection. The value '*Tall*' is not a test because *s*(*Tall*) = {3,4,5,7,8} and it is not equal to *splus*(*Tall*) = {4,5,7,8}. Also *t*(*splus*(*Tall*)) = '*Tall*' is not a test. The attributive projection of the value '*Tall*' on the set of positive examples is in Table 7.

Table - 6. The Example 2 of a Data Classification.

Index of Example	Height	Color of Hair	Color of Eyes	Class
1	Short	Blond	Blue	1
2	Short	Brown	Blue	1
3	Tall	Brown	Embrown	1
4	Tall	Blond	Embrown	2
5	Tall	Brown	Blue	2
6	Short	Blond	Embrown	2
7	Tall	Red	Blue	2
8	Tall	Blond	Blue	2

Table - 7. The Projection of the Value *'Tall'* on the Set *R*(+)*.*

Index of Example	Height	Color of Hair	Color of Eyes	Test?
4	Tall	Blond	Embrown	Yes
5	Tall	Brown	Blue	Yes
7	Tall		Blue	Yes
8	Tall	Blond	Blue	Yes

In this projection, *splus*(*Blue*) = {5,7,8}, *t*(*splus*(*Blue*)) = ‘*Tall Blue*’, *s*(*Tall Blue*) = {5,7,8} = splus(*Tall Blue*) hence ‘*Tall Blue*’ is a test for the second class. We have also that *splus*(*Brown*) = {5}, but {5} $\subseteq$ {5,7,8} and, consequently, there does not exist any good test which contains simultaneously the values ‘*Tall*‘ and ‘*Brown*’. Delete ‘*Blue*’ and ‘*Brown*’ from the projection as shown in Table 8.

However, now the rows t_5 and t_7 are not tests for the second class and they can be deleted as shown in Table 9. The intersection of the remaining rows of the projection is ‘*Tall Blond*’. We have that *s*(*Tall Blond*) = {4,8} $\subseteq$ *S*(+) and this collection of values is a test for the second class.

Table - 8. The Projection of the Value *‘Tall’* on *R(+)* without the Values *‘Blue’* and *‘Brown’.*

Index of Example	Height	Color of Hair	Color of Eyes	Test?
4	Tall	Blond	Embrown	Yes
5	Tall			No
7	Tall			No
8	Tall	Blond		Yes

Table - 9. The Projection of the Value *‘Tall’* on *R(+)* without the Examples t_5 and t_7.

Index of Example	Height	Color of Hair	Color of Eyes	Test?
4	Tall	Blond	Embrown	Yes
8	Tall	Blond		Yes

As we have found all the tests for the second class containing *‘Tall’* we can delete *‘Tall’* from the examples of the second class as shown in Table 10.

Table - 10. The Result of Deleting the Value *‘Tall’* from the Set *R(+).*

Index of Example	Height	Color of Hair	Color of Eyes	Test?	Class
1	Short	Blond	Blue	Yes	1
2	Short	Brown	Blue	Yes	1
3	Tall	Brown	Embrown	Yes	1
4		Blond	Embrown	Yes	2
5		Brown	Blue	No	2
6	Short	Blond	Embrown	Yes	2
7			Blue	No	2
8		Blond	Blue	No	2

Next we can delete the rows t_5, t_7, and t_8. The result is in Table 11.

The intersection of the remaining examples of the second class gives a test ‘*Blond Embrown*’ because

s(*Blond Embrown*) = *splus*(*Blond Embrown*) = {4,6} $\subseteq$ *S*(+).

Table - 11. The Result of Deleting t_5, t_7, and t_8 from the Set *R(+).*

Index of Example	Height	Color of Hair	Color of Eyes	Class
1	Short	Blond	Blue	1
2	Short	Brown	Blue	1
3	Tall	Brown	Embrown	1
4		Blond	Embrown	2
6	Short	Blond	Embrown	2

The choice of values or examples for forming a projection requires special consideration.

In contrast to incremental learning, where the problem is considered of how to choose relevant knowledge to be best modified, here we come across the opposite goal to eliminate irrelevant knowledge not to be processed.

Choosing Values and Examples for the Formation of Subtasks

Next, it is shown that it is convenient to choose essential values in an example and essential examples in a projection for the decomposition of the problem of inferring GMRTs into the subtasks of the first or second kind.

An Approach for Searching for Essential Values

Let t be a test for positive examples. Construct the set of intersections $\{t \cap t': t' \in R(-)\}$. It is clear that these intersections are not tests for positive examples. Take one of the intersections with the maximal number of values in it. The values complementing the maximal intersection in t is the minimal set of essential values in t.

Next we describe the procedure with the use of which a quasi-maximal subset of t^* that does not correspond to a test is obtained.

We begin with the first value a_1 t^*, then we take the next value a_2 of t^* and evaluate the function to_be_test ($\{a_1, a_2\}$). If the value of the function is *false*, then we take the next value a_3 of t^* and evaluate the function to_be_test ($\{a_1, a_2, a_3\}$)). If the value of the function to_be_test ($\{a_1, a_2\}$) is *true*, then the value a_2 of t^* is skipped and the function to_be_test ($\{a_1, a_3\}$)) is evaluated. We continue this process until we achieve the last value of t^*.

Return to Table 6. Exclude the value '*Red*' (we know that '*Red*' is a test for the second class) and find the essential values for the examples t_4, t_5, t_6, t_7, and t_8. The result is in Table 12.

Consider the value '*Embrown*' in t_6: *splus*(*Embrown*) = {4,6}, $t(\{4,6\})$ = '*Blond Embrown*' is a test.

The value '*Embrown*' can be deleted. But this value is only one essential value in t_6 and, therefore, t_6 can be deleted too. After that *splus*(*Blond*) is modified to the set {4,8}.

We observe that $t(\{4,8\})$ = '*Tall Blond*' is a test. Hence, the value '*Blond*' can be deleted from further consideration together with the row t_4. Now the intersection of the rows t_5, t_7, and t_8 produces the test '*Tall Blue*'.

Table - 12. The Essential Values for the Examples *t_4, t_5, t_6, t_7, and t_8.*

Index of Example	Height	Color of Hair	Color of Eyes	Essential Values	Class
1	Short	Blond	Blue		1
2	Short	Brown	Blue		1
3	Tall	Brown	Embrown		1
4	Tall	Blond	Embrown	Blond	2
5	Tall	Brown	Blue	*Blue, Tall*	2
6	Short	Blond	Embrown	Embrown	2
7	Tall		Blue	Tall, Blue	2
8	Tall	Blond	Blue	Tall	2

An Approach for Searching for Essential Examples

Let *STGOOD* be the partially ordered set of elements s satisfying the condition that $t(s)$ is a GMRT for $R(+)$. We can use the set *STGOOD* to find indices of essential examples in some subset s^* of indices for which $t(s^*)$ is not a test. Let $s^* = \{i_1, i_2, \dots , i_q\}$. Construct the set of intersections $\{s^* \cap s': s' \in STGOOD\}$. Any obtained intersection *corresponds* to a test for positive examples. Take one of the intersections with the maximal number of indices. The subset of s^* complementing in s^* the maximal intersection is the minimal set of indices of essential examples in s^*. For instance, $s^* = \{2,3,4,7,8\}$, $s' = \{2,3,4,7\}$, $s' \in STGOOD$, hence 8 is the index of essential example t_8 in s^*.

In the beginning of inferring GMRTs, the set *STGOOD* is empty. Next we describe the procedure with the use of which a quasi-maximal subset of s^* that corresponds to a test is obtained.

We begin with the first index i_1 of s^*, then we take the next index i_2 of s^* and evaluate the function to_be_test ($t(\{i_1, i_2\})$). If the value of the function is *tru*e, then we take the next index i_3 of s^* and evaluate the function to_be_test ($t(\{i_1, i_2, i_3\})$). If the value of the function to_be_test ($t(\{i_1, i_2\})$) is *false*, then the index i_2 of s^* is skipped and the function to_be_test ($t(\{i_1, i_3\})$) is evaluated. We continue this process until we achieve the last index of s^*.

For example, in Table 6, $S(+) = \{4,5,6,7,8\}$. Find the quasi-minimal subset of indices of essential examples for $S(+)$. Using the procedure described above we get that $t(\{4,6\})$ = '*Blond Embrown*' is a test for the second class and 5,7,8 are the indices of essential examples in S(+). Consider row t_5. We know that '*Blue*' is essential in it (see, please, Table 12). We have $t(splus(\{Blue\})) = t(\{5,7,8\})$ = '*Tall Blue*', and '*Tall Blue*' is a test for the second class of examples. Delete '*Blue*' and t_5. Now t_7 is not a test and we delete it. After that *splus*({*Tall*}) is modified to be the set {4,8}, and $t(\{4,8\})$ = '*Tall Blond*' is a test. Hence, the value '*Tall*' together with row t_8 cannot be considered for searching for new tests. Finally $S(+) = \{4,6\}$ corresponds to the test already known.

An Approach for Incremental Algorithms

The decomposition of the main problem of inferring GMRTs into subtasks of the first or second kind gives the possibility to construct incremental algorithms for this problem. The simplest way to do it consists of the following steps: choose example (value), form subproblem, solve subproblem (with the use of Algorithm 1 or Algorithm 2), delete example (value) after the subproblem is over, reduce *R*(+) and *T* and check the condition of ending the main task.

A recursive procedure for using attributive subproblems for inferring GMRTs has been described in [Naidenova et al., 1995b]. Some complexity evaluations of this algorithm can be found in [Naidenova and Ermakov, 2001]. In the following part of this chapter, we give an algorithm for inferring GMRTs the core of which is the decomposition of the main problem into the subtasks of the first kind combined with searching essential examples.

DIAGaRa: An Algorithm for Inferring All GMRTs with the Decomposition into Subtasks of the First Kind

The algorithm DIAGaRa for inferring all the GMRTs with the decomposition into subproblems of the first kind is briefly described in Figure 1.

The Basic Recursive Algorithm for Solving a Subtask of the First Kind

The initial information for the algorithm of finding all the GMRTs contained in a positive example is the projection of this example on the current set *R*(+). Essentially the projection is simply a subset of examples defined on a certain restricted subset *t** of values. Let *s** be the subset of indices of examples from *R*(+) which have produced the projection.

```
$s^* \leftarrow s(+) = \{1, \;\; , nt\}$;
$t^* \leftarrow T$;
Do
Begin
1.  to find all the GMRTs for a given set of positive examples with
    the use of the basic algorithm of solving subtask of the first
    kind;
End
```

Figure - 1. The Algorithm DIAGaRa.

It is useful to introduce the characteristic *W*(*t*) of any collection *t* of values named by the weight of *t* in the projection: $W(t) = ||s^* \cap s(t)||$ is the number of positive examples of the projection containing *t*. Let *WMIN* be the minimal permissible value of the weight.

Let *STGOOD* be the partially ordered set of elements *s* satisfying the condition that *t*(*s*) is a good test for *R*(+).

The basic algorithm consists of applying the sequence of the following steps:

Step 1. Check whether the intersection of all the elements of projection is a test and if so, then *s** is stored in *STGOOD* if *s** corresponds to a good test at the current step; in this case the subtask is over. Otherwise the next step is performed (we use the function to_be_test(*t*): if $s(t) \cap S(+) = s(t)$ ($s(t) \subseteq S(+)$) then *true* else *false*).

Step 2. For each value *A* in the projection, the set $splus(A) = \{s^* \cap s(A)\}$ and the weight $W(A) = ||splus(A)||$ are determined and if the weight is less than the minimum permissible weight *WMIN*, then the value *A* is deleted from the projection. We can also delete the value *A* if *W*(*A*) is equal to *WMIN* and *t*(*splus*(*A*)) is not a test – in this case *A* will not appear in a maximally redundant test *t* with *W*(*t*) equal to or greater than *WMIN*.

Step 3. The generalization operation is performed: $t' = t(splus(A))$, $A \in t^*$; if *t′* is a test, then the value *A* is deleted from the projection and *splus*(*A*) is stored in *STGOOD* if *splus*(*A*) corresponds to a good test at the current step.

Step 4. The value *A* can be deleted from the projection if $splus(A) \subseteq s'$ for some $s' \in STGOOD$.

Step 5. If at least one value has been deleted from the projection, then the reduction of the projection is necessary. The reduction consists of deleting the elements of projection that are not tests (as a result of previous eliminating values). If, under reduction, at least one element has been deleted from the projection, then Step 2, Step 3, Step 4, and Step 5 are repeated.

Step 6. Check whether the subtask is over or not. The subtask is over when either the projection is empty or the intersection of all elements of the projection corresponds to a test (see Step 1). If the subtask is not over, then the choice of an essential example in this projection is performed and the new subtask is formed with the use of this essential example. The new subsets *s** and *t** are constructed and the basic algorithm runs recursively. The important part of the basic algorithm is how to form the set *STGOOD.*

We give in the Appendix an example of the work of the algorithm DIAGaRa.

An Approach for Forming the Set *STGOOD*

Let $L(S)$ be the set of all subsets of the set S. $L(S)$ is the set lattice [Rasiova, 1974]. The ordering determined in the set lattice coincides with the set-theoretical inclusion. It will be said that subset s_1 is absorbed by subset s_2, i.e. $s_1 \leq s_2$, if and only if the inclusion relation is hold between them, i.e. $s_1 \subseteq s_2$. Under formation of *STGOOD*, a collection s of indices is stored in *STGOOD* if and only if it is not absorbed by any collection of this set. It is necessary also to delete from *STGOOD* all the collections of indices that are absorbed by s if s is stored in *STGOOD*. Thus, when the algorithm is over, the set *STGOOD* contains all the collections of indices that correspond to GMRTs and only such collections. Essentially the process of forming *STGOOD* is an incremental procedure of finding all maximal elements of a partially ordered set. The set *TGOOD* of all the GMRTs is obtained as follows: $TGOOD = \{t: t = t(s), (\forall s)\ (s \in STGOOD)\}$.

The Estimation of the Number of Subtasks to Be Solved

The number of subtasks at each level of recursion is determined by the number of essential examples in the projection associated with this level. The depth of recursion for any subtask is determined by the greatest cardinality (call it '*CAR*') of set-theoretical intersections of elements $s \in STGOOD$ corresponding to GMRTs: $CAR = \max\ (||s_i \cap s_j||, \forall(s_i, s_j)\ s_i, s_j \in STGOOD)$. In the worst case, the number of subtasks to be solved is of order $O(2^{CAR})$.

CASCADE: Inferring all GMRTs of Maximal Weight

The algorithm CASCADE serves for inferring all the GMRTs of maximal weight. At the beginning of the algorithm, the values are arranged in decreasing order of weight such that $W(A_1) \geq W(A_2) \geq \ldots \geq W(A_m)$, where $A_1, A_2, \ldots, A_m$ is a permutation of values. The shortest sequence of values $A_1, A_2, \ldots, A_j, j \leq m$ is defined such that it is a test for positive examples and *WMIN* is made equal to $W(A_j)$. The procedure DIAGaRa tries to infer all the GMRTs with weight equal to *WMIN*. If such tests are obtained, then the algorithm stops. If such tests are not found, then *WMIN* is decreased, and the procedure DIAGaRa runs again.

Conclusion

In this paper, we used a unified model for inferring implicative logical rules from examples. The key concept of our approach is the concept of a good diagnostic test. We define a good diagnostic test as the best approximation of a given classification on a given set of examples. In the framework of our approach, we show the equivalence between implicative rules and diagnostic tests for a given set of examples. The task of inferring good diagnostic tests from examples serves as an ideal model of inductive reasoning because this task realizes the canons of induction that has been originally formulated by English logician J.-S. Mill.

We have given the decomposition of inferring all good maximally redundant tests for a given set of examples into operations and subtasks that are in accordance with main human common sense reasoning operations. This decomposition allows, in principle, to transform the process of inferring good tests (and implicative rules) into a "step by step" reasoning process. Incremental algorithms of inferring good classification tests from examples demonstrate the possibility of this transformation in the best way.

We consider two kinds of subtasks: for a given set of positive examples 1) given a positive example *t*, find all GMRTs contained in *t*; 2) given a non-empty collection of values *X* (maybe only one value) such that it is not a test, find all GMRTs containing *X*. The decomposition of good classification tests inferring into subtasks implies introducing a set of special rules to realize the following operations: choosing examples (values) for subtasks, forming subtasks, deleting values or examples from subtasks and some other rules controlling the process of good test inferring. The concepts of an essential value and an essential example are introduced in order to optimize the choice of subtasks of the first and second kinds.

We have described an inductive algorithm DIAGaRa for inferring all good maximally redundant tests for a given set of positive examples. This algorithm realizes one of the possibilities to transform the searching of diagnostic tests (implicative logical rules) into “step by step” learning procedure.

Our approach is also applicable for inferring functional and associative dependencies from data.

Acknowledgements

The author is very grateful to Professor Evangelos Triantaphyllou (Louisiana StateUniversity) who inspired and supported this paper, and to Dr. Giovanni Felici (IASI – Italian National Research Council), for his invaluable advice concerning all the parts of this work.

Appendix

The data to be processed are in Table 13 (the set of positive examples) and in Table 14 (the set of negative examples).

An Example of Using the Algorithm DIAGaRa

We use the algorithm DIAGaRa for inferring all the GMRTs having the weight equal to or greater than *WMIN* = 4 for the training set of examples represented in Table 13 (the set of positive examples) and in Table 14 (the set of negative examples).

We begin with s^* = $S(+)$ = {{1}, {2}, …, {14}}, t^* = T = {A_1, A_2, ….., A_{26}}, *SPLUS* = {*splus*(A_i): $A_i \in t^*$} (see *SPLUS* in Table 15).

In table 15 and 16, A_* denotes the collection of values {A_8 A_9} and A_+ denotes the collection of values {A_{14} A_{15}} because *splus*(A_8) = *splus*(A_9) and *splus*(A_{14}) = *splus*(A_{15}).

Please observe that *splus*(A_{12}) = {2,3,4,7} and *t*({2,3,4,7}) is a test, therefore, A_{12} is deleted from t^* and *splus*(A_{12}) is inserted into *STGOOD*. Then *W*(A_*), *W*(A_{13}), and *W*(A_{16}) are less than *WMIN*, hence we can delete A_*, A_{13}, and A_{16} from t^*. Now t_{10} is not a test and can be deleted. After modifying *splus*(*A*) for A_5, A_{18}, A_2, A_3, A_4, A_6 A_{20}, A_{21}, and A_{26} we find that *W*(A_5) < *WMIN*, therefore, A_5 is deleted from t^* and *splus*(A_5) is inserted into *STGOOD*. Then *W*(A_{18}) turns out to be less than *WMIN* and we delete A_{18}, which implies deleting t_{13}. Next we modify *splus*(*A*) for A_1, A_{19}, A_{23}, A_4, A_{26} and find that *splus*(A_4) = {2,3,4,7}. A_4 is deleted from t^*. Finally, *W*(A_1) turns out to be less than *WMIN* and we delete A_1.

Table - 13. The Set of Positive Examples *R*(+).

Index of example	*R*(+)
1	A_1 A_2 A_5 A_6 A_{21} A_{23} A_{24} A_{26}
2	A_4 A_7 A_8 A_9 A_{12} A_{14} A_{15} A_{22} A_{23} A_{24} A_{26}
3	A_3 A_4 A_7 A_{12} A_{13} A_{14} A_{15} A_{18} A_{19} A_{24} A_{26}
4	A_1 A_4 A_5 A_6 A_7 A_{12} A_{14} A_{15} A_{16} A_{20} A_{21} A_{24} A_{26}
5	A_2 A_6 A_{23} A_{24}
6	A_7 A_{20} A_{21} A_{26}
7	A_3 A_4 A_5 A_6 A_{12} A_{14} A_{15} A_{20} A_{22} A_{24} A_{26}
8	A_3 A_6 A_7 A_8 A_9 A_{13} A_{14} A_{15} A_{19} A_{20} A_{21} A_{22}
9	A_{16} A_{18} A_{19} A_{20} A_{21} A_{22} A_{26}
10	A_2 A_3 A_4 A_5 A_6 A_8 A_9 A_{13} A_{18} A_{20} A_{21} A_{26}
11	A_1 A_2 A_3 A_7 A_{19} A_{20} A_{21} A_{22} A_{26}
12	A_2 A_3 A_{16} A_{20} A_{21} A_{23} A_{24} A_{26}
13	A_1 A_4 A_{18} A_{19} A_{23} A_{26}
14	A_{23} A_{24} A_{26}

Table - 14. The Set of Negative Examples $R(-)$.

Index of example	$R(-)$	Index of example	$R(-)$
15	$A_3\ A_8\ A_{16}\ A_{23}\ A_{24}$	32	$A_1\ A_2\ A_3\ A_7\ A_9\ A_{10}\ A_{11}\ A_{13}\ A_{18}$
16	$A_7\ A_8\ A_9\ A_{16}\ A_{18}$	33	$A_1\ A_5\ A_6\ A_8\ A_9\ A_{10}\ A_{19}\ A_{20}\ A_{22}$
17	$A_1\ A_{21}\ A_{22}\ A_{24}\ A_{26}$	34	$A_2\ A_8\ A_9\ A_{18}\ A_{20}\ A_{21}\ A_{22}\ A_{23}\ A_{26}$
18	$A_1\ A_7\ A_8\ A_9\ A_{13}\ A_{16}$	35	$A_1\ A_2\ A_4\ A_5\ A_6\ A_7\ A_9\ A_{13}\ A_{16}$
19	$A_2\ A_6\ A_7\ A_9\ A_{21}\ A_{23}$	36	$A_1\ A_2\ A_6\ A_7\ A_8\ A_{10}\ A_{11}\ A_{13}\ A_{16}\ A_{18}$
20	$A_{10}\ A_{19}\ A_{20}\ A_{21}\ A_{22}\ A_{24}$	37	$A_1\ A_2\ A_3\ A_4\ A_5\ A_6\ A_7\ A_{12}\ A_{14}\ A_{15}\ A_{16}$
21	$A_1\ A_{10}\ A_{20}\ A_{21}\ A_{22}\ A_{23}\ A_{24}$	38	$A_1\ A_2\ A_3\ A_4\ A_5\ A_6\ A_9\ A_{11}\ A_{12}\ A_{13}\ A_{16}$
22	$A_1\ A_3\ A_6\ A_7\ A_9\ A_{10}\ A_{16}$	39	$A_1\ A_2\ A_3\ A_4\ A_5\ A_6\ A_{14}\ A_{15}\ A_{19}\ A_{20}\ A_{23}\ A_{26}$
23	$A_2\ A_6\ A_8\ A_9\ A_{14}\ A_{15}\ A_{16}$	40	$A_2\ A_3\ A_4\ A_5\ A_6\ A_7\ A_{11}\ A_{12}\ A_{13}\ A_{14}\ A_{15}\ A_{16}$
24	$A_1\ A_4\ A_5\ A_6\ A_7\ A_8\ A_{11}\ A_{16}$	*41*	$A_2\ A_4\ A_5\ A_6\ A_7\ A_9\ A_{10}\ A_{11}\ A_{12}\ A_{13}\ A_{14}\ A_{15}\ A_{19}$
25	$A_7\ A_{10}\ A_{11}\ A_{13}\ A_{19}\ A_{20}\ A_{22}\ A_{26}$	42	$A_1\ A_2\ A_3\ A_4\ A_5\ A_6\ A_{12}\ A_{16}\ A_{18}\ A_{19}\ A_{20}\ A_{21}\ A_{26}$
26	$A_1\ A_2\ A_3\ A_5\ A_6\ A_7\ A_{10}\ A_{16}$	43	$A_4\ A_5\ A_6\ A_7\ A_8\ A_9\ A_{10}\ A_{11}\ A_{12}\ A_{13}\ A_{14}\ A_{15}\ A_{16}$
27	$A_1\ A_2\ A_3\ A_5\ A_6\ A_{10}\ A_{13}\ A_{16}$	44	$A_3\ A_4\ A_5\ A_6\ A_8\ A_9\ A_{10}\ A_{11}\ A_{12}\ A_{13}\ A_{14}\ A_{15}\ A_{18}\ A_{19}$
28	$A_1\ A_3\ A_7\ A_{10}\ A_{11}\ A_{13}\ A_{19}\ A_{21}$	45	$A_1\ A_2\ A_3\ A_4\ A_5\ A_6\ A_7\ A_8\ A_9\ A_{10}\ A_{11}\ A_{12}\ A_{13}\ A_{14}\ A_{15}$
29	$A_1\ A_4\ A_5\ A_6\ A_7\ A_8\ A_{13}\ A_{16}$	46	$A_1\ A_3\ A_4\ A_5\ A_6\ A_7\ A_{10}\ A_{11}\ A_{12}\ A_{13}\ A_{14}\ A_{15}\ A_{16}\ A_{23}\ A_{24}$
30	$A_1\ A_2\ A_3\ A_6\ A_{11}\ A_{12}\ A_{14}\ A_{15}\ A_{16}$	47	$A_1\ A_2\ A_3\ A_4\ A_5\ A_6\ A_8\ A_9\ A_{10}\ A_{11}\ A_{12}\ A_{14}\ A_{16}\ A_{18}\ A_{22}$
31	$A_1\ A_2\ A_5\ A_6\ A_{11}\ A_{14}\ A_{15}\ A_{16}\ A_{26}$	48	$A_2\ A_8\ A_9\ A_{10}\ A_{11}\ A_{12}\ A_{14}\ A_{15}\ A_{16}$

Table - 15. The Set *SPLUS* of the Collections *splus*(A) for all A in Tables 13 and 14.

$SPLUS = \{splus(A_i): s(A_i) \cap S(+), A_i \in T\}$:	
$splus(A_*) \rightarrow \{2,8,10\}$	$splus(A_{22}) \rightarrow \{2,7,8,9,11\}$
$splus(A_{13}) \rightarrow \{3,8,10\}$	$splus(A_{23}) \rightarrow \{1,2,5,12,13,14\}$
$splus(A_{16}) \rightarrow \{4,9,12\}$	$splus(A_3) \rightarrow \{3,7,8,10,11,12\}$
$splus(A_1) \rightarrow \{1,4,11,13\}$	$splus(A_4) \rightarrow \{2,3,4,7,10,13\}$
$splus(A_5) \rightarrow \{1,4,7,10\}$	$splus(A_6) \rightarrow \{1,4,5,7,8,10\}$
$splus(A_{12}) \rightarrow \{2,3,4,7\}$	$splus(A_7) \rightarrow \{2,3,4,6,8,11\}$
$splus(A_{18}) \rightarrow \{3,9,10,13\}$	$splus(A_{24}) \rightarrow \{1,2,3,4,5,7,12,14\}$
$splus(A_2) \rightarrow \{1,5,10,11,12\}$	$splus(A_{20}) \rightarrow \{4,6,7,8,9,10,11,12\}$
$splus(A_+) \rightarrow \{2,3,4,7,8\}$	$splus(A_{21}) \rightarrow \{1,4,6,8,9,10,11,12\}$
$splus(A_{19}) \rightarrow \{3,8,9,11,13\}$	$splus(A_{26}) \rightarrow \{1,2,3,4,6,7,9,10,11,12,13,14\}$

Table - 16. The sets *STGOOD* and *TGOOD* for the Examples of Tables 13 and 14.

№	*STGOOD*	*TGOOD*
1	2,3,4,7	$A_4\ A_{12}\ A_+\ A_{24}\ A_{26}$
2	1,2,12,14	$A_{23}\ A_{24}\ A_{26}$
3	4,6,8,11	$A_7\ A_{20}\ A_{21}$

We can delete also the values A_2, A_{19} because $W(A_2)$, $W(A_{19}) = 4$, $t(splus(A_2))$, $t(splus(A_{19}))$ are not tests and, therefore, these values will not appear in a maximally redundant test t with $W(t)$ equal to or greater than 4.

After deleting these values we can delete the examples t_9, t_5 because A_{19} is essential in t_9, and A_2 is essential in t_5. Next we can observe that $splus(A_{23}) = \{1,2,12,14\}$ and $t(\{1,2,12,14\})$ is a test, thus A_{23} is deleted from t^* and $splus(A_{23})$ is inserted into *STGOOD*. We can delete the value A_{22} and A_6 because $W(A_{22})$ and $W(A_6)$ are now equal to 4, $t(splus(A_{22}))$ and $t(splus(A_6))$ are not tests and these values will not appear in a maximally redundant test with weight equal to or greater than 4. Now t_{14} and t_1 are not tests and can be deleted.

Now choose t_6 as a subtask because this positive example is more difficult to be distinguished from the negative examples. By resolving this subtask, we find that t_6 produces a new test t with $s(t)$ equal to {4,6,8,11}. Delete t_6. We can also delete the value A_{21} because $W(A_{21})$ is now equal to 4, $t(splus(A_{21}))$ is not a test and this value will not appear in a maximally redundant test with weight equal to or greater than 4.

Now choose t_8 as a subtask because it belongs to the set of essential examples in the current projection with respect to the subset {2,3,4,7} that corresponds to one of the GMRTs already obtained. By resolving this subtask, we find that t_8 does not produce any new test. Delete $t_{8.}$ After that we can delete the values A_+, A_7, A_3, and A_{20} and these deletions imply than all of the remaining rows t_2, t_3, t_4, t_7, t_{11}, and t_{12} are not tests.

The list of all the GMRTs for the training set of positive examples is given in Table 16.

Bibliography

[Boldyrev, 1974]N. G. Boldyrev, “Minimization of Boolean Partial Functions with a Large Number of “Don’t Care” Conditions and the Problem of Feature Extraction”, *Proceedings of International Symposium “Discrete Systems”*, Riga, Latvia, pp.101-109, 1974.

[Cosmadakis et al., 1986] S. Cosmadakis, P. C. Kanellakis, N. Spyratos, “Partition Semantics for Relations”, *Journal of Computer and System Sciences*, Vol. 33, No. 2, pp.203-233, 1986.

[Demetrovics and Vu, 1993] J. Demetrovics and D. T. Vu, “Generating Armstrong Relation Schemes and Inferring Functional Dependencies from Relations”, *International Journal on Information Theory & Applications*, Vol. 1, No. 4, pp.3-12, 1993.

[Finn, 1984] V. K. Finn, “Inductive Models of Knowledge Representation in Man-Machine and Robotics Systems”, *Proceedings of VINITI*, Vol. A, pp.58-76, 1984.

[Ganascia, 1989] J.- Gabriel. Ganascia, “EKAW - 89 Tutorial Notes: Machine Learning”, *Third European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, Paris, France, pp. 287-296, 1989.

[Huntala et al., 1999] Y. Huntala, J. Karkkainen, P. Porkka, and H. Toivonen, “TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies”, *The Computer Journal*, Vol. 42, No. 2, pp. 100-111, 1999.

[Kuznetsov, 1993] S. O. Kuznetsov, “Fast Algorithm of Constructing All the Intersections of Finite Semi-Lattice Objects”, *Proceedings of VINITI*, Series 2, No. 1, pp. 17-20, 1993.

[Mannila and Räihä, 1992] H. Mannila, and K. – J. Räihä, “On the Complexity of Inferring Functional Dependencies”, *Discrete Applied Mathematics*, Vol. 40, pp. 237-243, 1992.

[Mannila and Räihä, 1994] H. Mannila, and K. – J. Räihä, “Algorithm for Inferring Functional Dependencies”. *Data & Knowledge Engineering*, Vol. 12, pp. 83-99, 1994.

[Megretskaya, 1989] I. A. Megretskaya, “Construction of Natural Classification Tests for Knowledge Base Generation”, in: *The Problem of the Expert System Application in the National Economy*, Kishinev, Moldavia, pp. 89-93, 1988.

[Mill, 1900] J. S. Mill, *The System of Logic*, Russian Publishing Company “Book Affair”: Moscow, Russia, 1900.

[Naidenova and Polegaeva, 1986] X. A. Naidenova, J. G. Polegaeva, “An Algorithm of Finding the Best Diagnostic Tests”, *The 4-th All Union Conference “Application of Mathematical Logic Methods”*, Theses of Papers, Mintz, G; E, Lorents, P. P. (Eds), Institute of Cybernetics, National Acad. of Sciences of Estonia, Tallinn, Estonia, pp. 63-67, 1986.

[Naidenova and Polegaeva, 1991] X. A. Naidenova, J. G. Polegaeva, “The System of Knowledge Acquisition from Experimental Facts”, in: *“Industrial Applications of Artificial Intelligence*”, James L. Alty and Leonid I. Mikulich (Eds), Elsevier Science Publishers B.V., Amsterdam, The Netherlands, pp. 87-92, 1991.

[Naidenova, 1992] X. A. Naidenova, “Machine Learning As a Diagnostic Task”, in: *“Knowledge-Dialogue-Solution”, Materials of the Short-Term Scientific Seminar*, Saint-Petersburg, Russia, editor Arefiev, I., pp.26-36, 1992.

[Naidenova et al., 1995a] X. A. Naidenova, J. G. Polegaeva, J. E. Iserlis, “The System of Knowledge Acquisition Based on Constructing the Best Diagnostic Classification Tests”, *Proceedings of International Conference “Knowledge-Dialog-Solution”*, Jalta, Ukraine, Vol. 1, pp. 85-95, 1995a.

[Naidenova et al., 1995b] X. A. Naidenova, M. V. Plaksin, V. L. Shagalov, “Inductive Inferring All Good Classification Tests”, *Proceedings of International Conference “Knowledge-Dialog-Solution*”, Jalta, Ukraine, Vol. 1, pp.79-84, 1995b.

[Naidenova, 1996] X. A. Naidenova, “Reducing Machine Learning Tasks to the Approximation of a Given Classification on a Given Set of Examples”, *Proceedings of the 5-th National Conference at Artificial Intelligence*, Kazan, Tatarstan, Vol. 1, pp. 275-279, 1996.

[Naidenova, 1999] X. A. Naidenova, “The Data-Knowledge Transformation”, in: “*Text Procesing and Cognitive Technologies”, Paper Collection*, editor Solovyev, V. D., - Pushchino, Russia, Vol. 3, pp. 130-151, 1999.

[Naidenova and Ermakov, 2001] X. A. Naidenova, A. E. Ermakov, “The Decomposition of Algorithms of Inferring Good Diagnostic Tests”, *Proceedings of the 4-th International Conference “Computer – Aided Design of Discrete Devices” (CAD DD’2001)*, Institute of Engineering Cybernetics, National Academy of Sciences of Belarus, editor A. Zakrevskij, Minsk, Belarus, Vol. 3, pp. 61-69, 2001.

[Naidenova, 2001] X. A. Naidenova, “Inferring Good Diagnostic Tests as a Model of Common Sense Reasoning”, *Proceedings of the International Conference “Knowledge-Dialog-Solution” (KDS’2001)*, State North-West Technical University, Publishing House « Lan », Saint-Petersburg, Russia, Vol. II, pp. 501-506, 2001.

[Ore, 1944] O. Ore, “*Galois Connexions*”, Trans. Amer. Math. Society, Vol. 55, No. 1, pp. 493-513, 1944.

[Piaget, 1959] J.Piaget, *La genèse des Structures Logiques Elémentaires*, Neuchâtel, 1959.

[Riguet, 1948] J. Riguet, “Relations Binaires, Fermetures, Correspondences de Galois”, *Bull. Soc. Math.*, France, Vol. 76., No 3, pp.114-155, 1948.

[Shreider, 1974] J. Shreider, “Algebra of Classification”, *Proceedings of VINITI*, Series 2, No. 9, pp. 3-6, 1974.

[Sperner, 1928] E. Sperner, “Eine satz uber Untermengen einer Endlichen Menge”. *Mat. Z.*, Vol. 27, No. 11, pp. 544-548, 1928.

[Wille, 1992] R. Wille, “Concept Lattices and Conceptual Knowledge System”, *Computer Math. Appl.*, Vol. 23, No. 6-9, pp. 493-515, 1992.

Author's Information

Naidenova Xenia Alexandrovna - Military medical academy, Saint-Petersburg, Stoikosty street, 26-1-248, naidenova@mail.spbnit.ru.

ACTIVE MONITORING AND DECISION MAKING PROBLEM

Sergey Mostovoi, Vasiliy Mostovoi

Abstract: Active monitoring and problem of non-stable of sound signal parameters in the regime of piling up response signal of environment is under consideration. Math model of testing object by set of weak stationary dynamic actions is offered. The response of structures to the set of signals is under processing for getting important information about object condition in high frequency band. Making decision procedure by using researcher’s heuristic and aprioristic knowledge is discussed as well. As an example the result of numerical solution is given.

Keywords: math model, active monitoring, set of weak stationary dynamic actions.

ACM Classification Keywords: I.6.1 Simulation Theory.

Introduction

The distinctive feature of seismic monitoring is the particular, seismic frequency range, encompassing infrasonic and low range of a sound spectrum. The characteristics of each monitoring object are slowly varied in time, but at the same time sometimes processes might be occurred is too rapid. The seismic monitoring deals with the large size objects, down to the sizes of a terrestrial Globe. Because of mankind anxiety on possible earthquakes, the extremely passive monitoring has a deep history, but at latest time the active monitoring is often used. The active monitoring is such an experiment, which one is connected to generation of sounding signal of a different type, both on a spectral band, and on duration and power, down to atomic explosions. But in active experiment only monitoring approach enables to obtain ecological pure result, i.e. without any of appreciable influencing on an environment. Monitoring is a set of regime observations, and condition of observations and the characteristics of sounding signal depend on the purposes of given investigation. There are many such purposes, but, from our point of view, we select two basic one. It is dynamics of variations happening in investigated object, and it is detail of estimations, which characterise this object. Despite of large discrepancy of these two purpose, the approaches both to experimentation and to processing receivable data are very close, as well as problems, originating at it.

To problems, first of all from the ecological point of view, it is necessary to refer necessity to realize active monitoring of investigated object by low-power signals, commensurable with a level of a natural background. This circumstance results that the estimation of sounding signal parameters, passing the studied object, i.e. signal response of an investigated system on a sounding signal, is hampered because of a low signal-noise proportion.