247/2008

Raport Badawczy

Research Report

RB/49/2008

An integer programming approach to inductive learning from examples

J.Kacprzyk, G.Szkatuła

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences

POLSKA AKADEMIA NAUK

Instytut Badań Systemowych

ul. Newelska 6

01-447 Warszawa

tel.: (+48) (22) 3810100

fax: (+48) (22) 3810105

Kierownik Pracowni zgłaszający pracę:
Prof. dr hab. inż. Janusz Kacprzyk

Warszawa 2008

An Integer Programming Approach to Inductive Learning From Examples

1. Introduction

Machine learning from examples is a process of inferring a classification rule (concept description) of a class from descriptions of some individual elements of the class called *positive examples*, with some elements from outside of the class, called *negative examples,* which are used for narrowing the solution space. Each example is assumed positive or negative.

Traditionally, the most relevant requirements to be satisfied by learning procedures are:

- *completeness*, i.e. that the classification rule must correctly describe *all the* positive examples,
- *consistency*, i.e. that the classification rule must not describe *any* negative examples,
- *convergence*, i.e. the classification rule must be derived in a *finite* number of steps,
- the classification rule of *minimal "length*" is to be found, e.g. with the minimum number of attributes (or, more generally being "simple").

The sense of the first three is quite natural, and the sense of the fourth reflects an obvious fact that long rules are not "legible" to the humans; hence their practical usefulness may be limited.

In practice, due to imperfect data and other elements of the process, the first two requirements are usually meant in a relaxed way as:

- *a partial completeness*, i.e. that the classification rule must correctly describe, say, *most* of the positive examples,
- *a partial consistency*, i.e. that the classification rule must describe, say, *almost none* of the negative examples.

Examples are described (cf. Michalski, 1983) by a set of *K* "attribute - value" pairs written as

$$e = \bigwedge_{j=1}^{K} [a_j \# v_j]$$

where a_j denotes attribute *j* with value v_j and # is a relation exemplified by =, <, > etc.

For instance, if the attributes are: *height, color_of_hair, color_of_eyes*, than the concept "look of a one women" may be described by

[height = "high"]∧[color_of_hair = "blond"]∧[color_of_eyes = "blue"].

We propose here a modified inductive learning procedure based on Michalski's (1973, 1983) star-type methodology. The method is based on some elements of the authors' previous work (cf. Kacprzyk and Szkatuła, 1994a,b, 1995, 1996, 1997a,b, 1998, 1999, 2002a,b, 2005a,b,c and Szkatuła and Kacprzyk, 2005). Basically, since the algorithm builds the rule sought in an iterative way, a pre-processing of data (examples) is performed based on an analysis of how frequent the values of the particular attributes occur in the examples. These frequencies are used to derive weights associated with those values, and the problem is represented as a modification of the set covering problems, and solved by a modification of a greedy algorithm (IP_GRE) or a genetic algorithm (IP_GA).

The paper is organized as follows. In Section 2 the inductive learning problem is represented as a modification of the set covering problems. In Section 3 the basic steps of the IP method are described. In Section 4 the IP_GRE procedure is presented. In Section 5 the IP_GA procedure is presented. In Section 6 and Section 7 computation results are given.

2. Problem Formulation of Inductive Learning from Examples

Suppose that we have a finite set of examples U and a finite set of attributes $A=\{a_1,\ldots,a_K\}$. $V_{a_j}=\{v_{j,1},v_{j,2},\ldots,v_{j,L_j}\}$ is a domain of the attribute a_j, $j=1,\ldots,K$, where L_j denotes number of values of the j-th attribute. $V=\bigcup_{j=1,\ldots,K} V_{a_j}$. $f:U\times A\rightarrow V$ is a function such that $f(e^n,a_j)\in V_{a_j}$ for every $a_j\in A$ and $e^n\in U$, $n=1,2,\ldots,N$. Each example $e^n\in U$ is described by K attributes $A=\{a_1,\ldots,a_K\}$, and is represented by

$$e^n=\bigwedge_{j=1}^{K}[a_j=f(e^n,a_j)] \tag{1}$$

where $f(e^n,a_j)=v_{j,t(j,n)}$, and $v_{j,t(j,n)}\in V_{a_j}$. Function $f(e^n,a_j)$ denotes that the attribute a_j taking on a value $v_{j,t(j,n)}$ for example e^n. The index $t(j,n)$ for $j\in\{1,2,\ldots,K\}$ and $n\in\{1,2,\ldots,N\}$ specifies which value of the j-th attribute is used in the n-th example.

An example e^n in (1) is composed of K "attribute-value" pairs (*selectors*), denoted $s_j=[a_j=v_{j,t(j,n)}]$. Conjunction of $l\leq K$ "attribute-value" pairs, i.e.

$$\bigwedge_{j\in I} s_j=C^l \tag{2}$$

where $I\subseteq\{1,\ldots,K\}$, $card(I)=l$ is called a *complex*.

A complex C^l *covers* an example e^n if all the conditions on attributes given as j- th selectors are covered by (equal to) the values of the respective attributes in the example, $\forall j\in I$. The set of all the examples described by the conjunction C^l will be denoted $[C^l]$.

For instance, for *K=3* the complex [a_1= "woman"]∧[a_3 = "35 years"] covers the example [a_1= "woman"]∧[a_2= "married"]∧[a_3= "35 years"] but does not cover the example [a_1 = "man"]∧[a_2= "married"]∧[a_3= "35 years"].

Suppose that we have a finite set of examples *U*, a finite set of attributes $A \cup \{a_d\}$, $\{a_d\} \cap A = \emptyset$, a_d is a decision attribute and $V_{a_d} = \{v_{d,1}, v_{d,2}, \ldots, v_{d,L_d}\}$ is a domain of the attribute a_d. We have the sets $\{Y_{v_{d,l}} : l = 1,\ldots, L_d\}$, where $Y_{v_{d,l}} = \{e \in U : f(e,a_d) = v_{d,l}\}$, $\forall v_{d,l} \in V_{a_d}$ and $Y_{v_{d,1}} \cup \ldots \cup Y_{v_{d,L_d}} = U$, $Y_{v_{d,i}} \cap Y_{v_{d,j}} = \emptyset$ for $i \neq j$. Thus, the decision attribute splits the set of examples into the non-empty, disjoint and exhaustive subsets, that we call the *decision classes*.

Let us class $Y_{v_{d,l}}$, for $v_{d,l} \in V_{a_d}$. Suppose that we have a set of *positive examples*:

$$S_P(Y_{v_{d,l}}) = \{e \in U : f(e,a_d) = v_{d,l}\} \tag{3}$$

and a set of *negative examples*:

$$S_N(Y_{v_{d,l}}) = \{e \in U : f(e,a_d) \neq v_{d,l} \text{ and } \forall e' \in S_P(Y_{v_{d,l}})\ \exists a_j \in A, f(e,a_j) \neq f(e',a_j)\} \tag{4}$$

with $S_P(Y_{v_{d,l}}), S_N(Y_{v_{d,l}}) \neq \emptyset$, $S_P(Y_{v_{d,l}}) \cap S_N(Y_{v_{d,l}}) = \emptyset$, by assumption.

An implication R_k: $C^{I_k} \to [a_d = v_{d,l}]$, $l \in \{1,\ldots,L_d\}$ is called the *k*-th *"elementary" rule* for the class $Y_{v_{d,l}}$, where $C^{I_k} = \underset{j \in I_k}{\wedge} [a_j = v_{j,t(j,k)}]$ is description of example in terms of condition attributes $a_j, j \in I_k$, $I_k \subseteq \{1,\ldots,K\}$ and this example belongs to class $Y_{v_{d,l}}$. The index $t(j,k)$ specifies which value of the *j*-th attribute is used in the *k*-th rule.

Each rule is characterised by the coefficient of its strength. The strength of a rule R_k, which depends upon the number of examples described by the conditional part of the rule C^{I_k}, belonging to a given class $Y_{v_{d,l}}$ is defined in the following manner: $q(C^{I_k}) = \dfrac{card(\{e : e \in [C^{I_k}] \text{ and } f(e,a_d) = v_{d,l}\})}{card(\{e : e \in U\})}$. It is evident that $0 \leq q(C^{I_k}) \leq 1$. The more examples are described by the rule, the greater the value of the rule strength coefficient (i.e. the more important the rule is).

In this paper we consider the classification rules to be the disjunction (via "∪") of "elementary" rules consisting of complexes of type (2), i.e.

$$C^{I_1} \cup \ldots \cup C^{I_L} = \underset{j_1 \in I_1}{\wedge} s_{j_1} \cup \ldots \cup \underset{j_L \in I_L}{\wedge} s_{j_L} \to [\text{class} = \text{class } Y_{v_{d,l}}] \tag{5}$$

where: $I_1, \ldots, I_L \subseteq \{1, \ldots, K\}$ and "∪" corresponds to the connective "or".

Suppose that we have a finite set of examples U, a finite set of attributes $\{a_1,\ldots,a_K\}\cup\{a_d\}$, a_d is a decision attribute and $V_{a_d}=\{v_{d,1},v_{d,2},\ldots,v_{d,L_d}\}$ is a domain of the attribute a_d. Let us class $Y_{v_{d,l}}$, for $v_{d,l}\in V_{a_d}$. Suppose now that we have P positive examples and N negative examples. The positive examples are therefore $e^m\in S_P(Y_{v_{d,l}})$, $m=1,\ldots,P$, and are written as

$$e^m=[a_1=v_{1,t(1,m)}]\wedge\ldots\wedge[a_K=v_{K,t(K,m)}] \tag{6}$$

where: $v_{j,t(j,m)}\in V_{a_j}$, $j=1,\ldots,K$, while the negative examples are $e^n\in S_N(Y_{v_{d,l}})$, $n=1,\ldots,N$, and are written as

$$e^n=[a_1=v_{1,t(1,n)}]\wedge\ldots\wedge[a_K=v_{K,t(K,n)}] \tag{7}$$

where: $v_{j,t(j,n)}\in V_{a_j}$, $j=1,\ldots,K$ and $S_P(Y_{v_{d,l}}),S_N(Y_{v_{d,l}})\neq\varnothing$, $S_P(Y_{v_{d,l}})\cap S_N(Y_{v_{d,l}})=\varnothing$.

The idea of data pre-processing proposed in this work is as follows. For each attribute a_j in the examples, it is evident that not each possible value occurs at the same intensity (frequency). Clearly, if a value occurs more frequently in the positive examples and less frequently in the negative examples, then it can be argued that it should rather appear in the rule sought. Clearly, these frequencies should be relative because the sizes of sets of the positive and negative examples need not be the same.

The above rationale may be formalized as follows: first we introduce the following function, for each attribute a_j, $j=1,\ldots,K$ and particular values $v_{j,l}\in V_{a_j}$, $l=1,\ldots,L_j$, where the j-th attribute a_j taking on the value $v_{j,t(j,m)}$ for example $e^m\in S_P$ and the value $v_{j,t(j,n)}$ for example $e^n\in S_N$

$$g_j(v_{j,l})=\frac{1}{P}\sum_{m=1}^{P}\delta(e^m,v_{j,l})-\frac{1}{N}\sum_{n=1}^{N}\delta(e^n,v_{j,l}) \tag{8}$$

where:

$$\delta(e^m,v_{j,l})=\begin{cases}1 & \mathit{for}\ v_{j,t(j,m)}=v_{j,l}\\ 0 & \mathit{otherwise}\end{cases}$$

$$\delta(e^n,v_{j,l})=\begin{cases}1 & \mathit{for}\ v_{j,t(j,n)}=v_{j,l}\\ 0 & \mathit{otherwise}\end{cases}$$

Therefore, (8) expresses to what degree the particular values $v_{j,l}\in V_{a_j}$ of attribute a_j occurs more often in the positive than in the negative examples. We may therefore assume that this normalized value $g_j(v_{j,l})$ is used as a weight of value $v_{j,l}\in V_{a_j}$ of each attribute a_j, due to the rationale mentioned above.

Suppose now that we have a positive example e^m, such that $e^m=[a_1=v_{1,t(1,m)}]\wedge\ldots\wedge[a_K=v_{K,t(K,m)}]$, and we consider a complex

$$C^I = [a_{j_1} = v_{j_1,t(j_1,m)}] \wedge \ldots \wedge [a_{j_m} = v_{j_m,t(j_m,m)}] \tag{9}$$

that corresponds to the set of indices $I = \{j_1, \ldots, j_m\} \subseteq \{1, \ldots, K\}$; the set of indices $\{j_1, \ldots, j_m\}$ is clearly equivalent to a vector $x = [x_j]^T$, such that $x_j = 1$ if a selector $[a_j = v_{j,t(j,m)}]$ occurs in the complex (9), and 0 otherwise, $j = 1, \ldots, K$.

For instance, for $K = 3$ and example e^m = [height = "high"]$\wedge$[color_of_hair = "blond"]$\wedge$ [color_of_eyes = "blue"], the vector $[0,1,0]^T$ is equivalent to the complex [color_of_hair = "blond"]; the complex [height = "high"] is equivalent to the vector $[1,0,0]^T$.

As already mentioned, to the value $v_{j,t(j,m)} \in V_{a_j}$, of each attribute a_j a weight $g_j(v_{j,t(j,m)})$ is assigned, and an example e_W^m with weights is written as

$$e_W^m = \bigwedge_{j=1}^{K} [a_j = v_{j,t(j,m)}; g_j(v_{j,t(j,m)})] \tag{10}$$

An example e_W^m given by (10) is then composed of the weighted selectors, $s_j^W = [a_j = v_{j,t(j,m)}; g_j(v_{j,t(j,m)})]$, and a conjunction of them, i.e.

$$\begin{aligned} C_W^I &= \bigwedge_{j \in I \subseteq \{1,\ldots,K\}} s_j^W = \\ &= \bigwedge_{j=j_1,j_2,\ldots,j_m} s_j^W = [a_{j_1} = v_{j_1,t(j_1,m)}; g_{j_1}(v_{j_1,t(j_1,m)})] \wedge \ldots \wedge [a_{j_m} = v_{j_m,t(j_m,m)}; g_{j_m}(v_{j_m,t(j_m,m)})] \end{aligned} \tag{11}$$

is called a *weighted complex*. Notice that for the above C_W^I the vector x has the elements $x_{j_1}, x_{j_2}, \ldots, x_{j_m} = 1$, while, for $j \in \{1,2,\ldots,K\} \setminus \{j_1, j_2, \ldots, j_m\}$ we have $x_j = 0$.

For a weighted complex C_W^I its *weighted length* is

$$d_W(C_W^I) = \sum_{j=j_1}^{j_m} (1 - g_j(v_{j,t(j,m)})) \cdot x_j + \sum_{j \in \{1,\ldots,K\} \setminus \{j_1,j_2,\ldots,j_m\}} (1 - g_j(v_{j,t(j,m)})) \cdot x_j = \sum_{j=1}^{K} (1 - g_j(v_{j,t(j,m)})) \cdot x_j \tag{12}$$

which reflects the philosophy of data pre-processing introduced above, i.e. a higher relevance of those values of attributes which occur more often in positive than in negative examples.

The *length of the weighted classification rule* composed of L weighted complexes, $R_W \to [class = Y_{v_{d,j}}]$, $R_W = C_W^{I_1} \cup \ldots \cup C_W^{I_L}$ is

$$d_{R_W}(C_W^{I_1} \cup \ldots \cup C_W^{I_L}) = \max_{i=1,\ldots,L} d_W(C_W^{I_i}) \tag{13}$$

The problem of learning from examples was to find an optimal classification rule $R_W^* = C_W^{I_1^*} \cup \ldots \cup C_W^{I_L^*}$ such that

$$\min_{I_1,\ldots,I_L} d_{R_W}(C_W^{I_1} \cup \ldots \cup C_W^{I_L}) \tag{14}$$

i.e. which minimizes the weighted length of the classification rule.

Since the (exact) solution of problem (14) is very difficult, an auxiliary problem is solved (cf. Kacprzyk and Szkatula, 1996), i.e. an $R_W^* = C_W^{I_1 *} \cup .. \cup C_W^{I_L *}$ is sough such that

$$\min_{I_1} d_W(C_W^{I_1}), \ldots, \min_{I_L} d_W(C_W^{I_L}) \tag{15}$$

where the minimization is consecutively performed over the sets of indices $I_1, \ldots, I_L$ [cf. (5)]; the solution of (15) is in general very close to that of (14), while much easier to obtain.

If the requirements to be satisfied by learning procedures are:

- *a partial completeness*, i.e. that the classification rule must correctly describe *most* of the positive examples,
- *a partial consistency*, i.e. that the classification rule must not describe *most* of the negative examples,

the problem (15) can be represented as a modification of the set covering problem (SCP).

3. Solution by Using the IP Method

Suppose that we have a finite set of examples *U*, a finite set of attributes $\{a_1, \ldots, a_K\} \cup \{a_d\}$, a_d is a decision attribute and $V_{a_d} = \{v_{d,1}, \ldots, v_{d,L_d}\}$ is a domain of the attribute a_d. We have the decision classes $\{Y_{v_{d,l}} : l = 1, \ldots, L_d\}$, where $Y_{v_{d,l}} = \{e \in U : f(e, a_d) = v_{d,l}\}$, $\forall v_{d,l} \in V_{a_d}$. The rules are iteratively induced for each class.

Let us class $Y_{v_{d,l}}$ for $v_{d,l} \in V_{a_d}$. Suppose that we have a set of positive examples S_P [cf. (3)], $e^m \in S_P$, $m = 1, \ldots, P$ and a set of negative examples S_N [cf. (4)], $e^{P+n} \in S_N$, $n = 1, \ldots, N$, $S_N, S_P \neq \varnothing$, $S_P \cap S_N = \varnothing$, by assumption.

For example $e^m \in S_P$, where

$$e^m = [a_1 = v_{1,t(1,m)}; g_1(v_{1,t(1,m)})] \wedge .. \wedge [a_K = v_{K,t(K,m)}; g_K(v_{K,t(K,m)})]$$

and all the negative examples $e^{P+n} \in S_N$, $n = 1, \ldots, N$, where

$$e^{P+n} = [a_1 = v_{1,t(1,P+n)}; g_1(v_{1,t(1,P+n)})] \wedge \ldots \wedge [a_K = v_{K,t(K,P+n)}; g_K(v_{K,t(K,P+n)})],$$

we construct a 0-1 matrix $Z_{N*K} = [z_{nj}]$, $n = 1, \ldots, N$, $j = 1, \ldots, K$, defined as

$$z_{nj} = \begin{cases} 1 & for \quad v_{j,t(j,m)} \neq v_{j,t(j,P+n)} \\ 0 & for \quad v_{j,t(j,m)} = v_{j,t(j,P+n)} \end{cases} \tag{16}$$

The rows of this matrix correspond to the consecutive negative examples $e^{P+n} \in S_N$, $n = 1, \ldots, N$ and the columns correspond to the subsequent attributes $a_1, \ldots, a_K$. The value $z_{nj} = 1$ occurs if attribute a_j takes on different values in the positive and negative example, i.e. $v_{j,t(j,m)}$ in e^m are not equal to $v_{j,t(j,P+n)}$ in e^{P+n}; and $z_{nj} = 0$ otherwise.

In such a matrix there are clearly no rows with all the elements equal 0 since, by assumption, the sets of positive and negative examples are disjoint (and non-empty), $S_N, S_P \neq \emptyset$, $S_P \cap S_N = \emptyset$. Thus, for any positive and negative example there always exists at least one attribute taking on a different value in these examples.

Consider now the following inequality

$$\sum_{j=1}^{K} z_{nj} x_j \geq \gamma_n , \qquad n = 1, \ldots, N \tag{17}$$

where $\gamma = [\gamma_1, \ldots, \gamma_N]^T$ is a zero-one vector (of N elements), and $x = [x_1, \ldots, x_K]^T$ such that $x_j \in \{0,1\}$, for $j = 1, \ldots, K$ [cf. the remark in (9)].

Any vector x defined above which satisfies inequality $Zx \geq \gamma$ (17) determines therefore in a unique way [cf. (9)] some complex composed of selectors from the description of the example such that the conditions of partial completeness and partial consistence are satisfied. It describes at least one example from the set of positive examples, and it does not describe most of the examples from the set of negative examples. If vector x does not describe the n-th negative example, than $\gamma_n = 1$; and $\gamma_n = 0$ otherwise.

The minimization in problem (15) may be written using the inequality (17) as

$$\min_{x:Zx \geq \gamma} \sum_{j=1}^{K} (1 - g_j(v_{j,t(j,m)})) \cdot x_j \tag{18}$$

The minimization over the set of indices I_l may be replaced by the minimization with respect to x which yields [cf. (14)] an $R_W^* = C_W^{I_1^*} \cup \ldots \cup C_W^{I_L^*}$ such that

$$\min_{x:Z^1 x \geq \gamma} d_W(C_W^{I_1^*}), \ldots, \min_{x:Z^L x \geq \gamma} d_W(C_W^{I_L^*}) \tag{19}$$

Each minimization with respect to x in (19) is therefore equivalent to the determination of a 0-1 vector x^* which uniquely determines the complex of the shortest weighted length. On the other hand, the satisfaction of $Zx \geq \Lambda$ (if Λ is a unit vector) guarantees that such a complex would not describe the all negative examples. If a complex must describe *almost none* of the negative examples, problem (18) can be written as a modification of the set covering problem (SCP)

$$\min_{x,\gamma} \sum_{j=1}^{K} c_j x_j \tag{20}$$

subject to

$$\sum_{j=1}^{K} z_{nj} x_j \geq \gamma_n \,, \qquad n = 1, \ldots, N \tag{21}$$

and an additional constraint

$$\sum_{n=1}^{N} \gamma_n \geq N - rel \tag{22}$$

where

$$\begin{gathered} c_j = (1 - g_j(v_{j,t(j,m)})) \,, \quad z_{nj} \in \{0,1\} \,, \quad x_j \in \{0,1\} \,, \\ j = 1, \ldots, K \,, \quad \gamma = [\gamma_1, \ldots, \gamma_N]^T \,, \quad \gamma_n \in \{0,1\} \,, \quad n = 1, \ldots, N \end{gathered} \tag{23}$$

given a parameter $rel \geq 0$.

The above problem is the same as the original SCP with the exception that no more then *rel* rows are uncovered. Then, it is clear that no more then *rel* rows can be deleted from the problem. Note here that we may, in deleting rows, lose some information about the problem that could have been better used. Note also that this reduction test cannot always be applied. In the set covering problem (SCP) [cf. Beasley and Chu (1996)] there is only constrains (21) and vector $\gamma = [\gamma_1, \ldots, \gamma_N]^T$ is a unit vector.

Our modified set covering problem (20), (21), (22), (23) is the problem of covering at least *N-rel* rows of an *N*-row, *K*-column, zero-one matrix (z_{nj}) by a subset of the columns at minimal cost c_j. We define $x_j = 1$ if column *j* with cost $c_j > 0$ is in the solution and $x_j = 0$ otherwise. Equations (21) and (22) ensures that the most rows (at least *N-rel* rows) are covered by at least one column and equation (23) is the integrally constraint. It always has a feasible solution (a unit vector *x* of *K* element), due to the required disjoints of the sets of positive and negative examples and the way the matrix *Z* was constructed.

So, we are looking for a 0-1 vector *x* at minimum cost and a 0-1 vector $\gamma = [\gamma_1, \ldots, \gamma_N]^T$ which determines the covered rows, $\gamma_n = 1$ if *n*-th row is covered by solution *x* and $\gamma_n = 0$ otherwise. By assumption, at least *N-rel* rows (given a parameter $rel \geq 0$) must be covered by solution *x*.

Then, an "elementary" rule for class $Y_{v_{d,l}}$, $v_{d,l} \in V_{a_d}$, may not describe at least $(\frac{100}{N}\sum_{n=1}^{N}\gamma_n)$% negative examples.

Example 1. Let consider the class $Y_{v_{d,l}}$, for $v_{d,l} \in V_{a_d}$. Suppose that in Fig. 1 all the training examples are shown. Those belonging to the class (the six positive examples) are marked by $\oplus$ and those not belonging to the class (the six negative examples) are marked by $\ominus$.

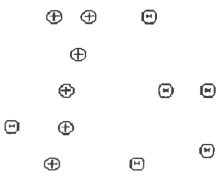


Figure 1.

A one elementary rule for the class $Y_{v_{d,l}}$, for *rel* = 0 (that not describe all the negative examples) and for *rel* = 1 (i.e. may don't describe no more than one training example), is illustrated in Fig. 2.

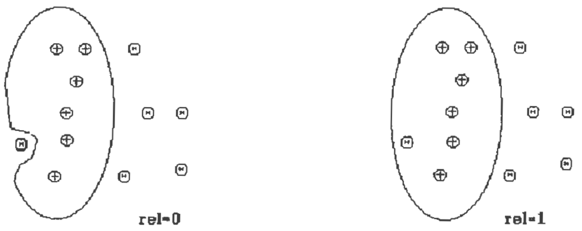


Figure 2.

□

The set covering problem is a well-known combinatorial optimization problem and has been proven to be NP-complete (Garey and Johnson, 1979). NP-complete problems are problems that are not currently solvable in polynomial time. A number of optimal and heuristic algorithms which try to find a "good" solution quickly have been presented in the literature in recent years.

The first published approximation algorithms for the SCP with a worst-case analysis used the greedy heuristic [cf. Johnson (1974), Lovasz (1975), Chvatal (1979)]. The approximation ratio of greedy algorithm (i.e. the worst ratio between the cost of a greedy solution and the optimum) is $\ln N + 1$. A probabilistic analysis of the SCP defined by randomly generated matrices appears in Vercellis (1984).

A number of optimal algorithms for the SCP, typically based upon tree-search procedures, have appeared in Balas and Ho (1980), and Beasley (1987). Beasley (1987) presented an algorithm for the SCP that combines problem reduction tests with dual ascent, subgradient optimization and linear programming. Fisher and Kedia (1990) presented an optimal solution algorithm based on a dual heuristic. Beasley (1992) combined a Lagrangian heuristics, feasible solution exclusion constraints, Gomory's f-cuts and an improved branching strategy to enhance his previous algorithm [cf. Beasley (1987)]. Harche and Thomson (1994) developed an exact algorithm based on a new method, called a column subtraction (or row sum) method, which is capable of solving large sparse instances of set covering problems.

Among the heuristic methods, Balas and Ho (1980) reported that a reasonable lower bound for the SCP could be found by a dual ascent procedure together with subgradient optimization to improve upon the lower bound obtained from the dual ascent procedure. They considered a heuristic of the greedy type where at each stage an uncovered row *i* is chosen and covered by choosing the column j which minimizes some function F (c_j, number of uncovered rows in column *j*, *j* covers *i*) to generate an upper bound (Z_{UB}) for the problem. Balas and Ho considered five different forms for the function F. A key feature of their work was the use cutting planes. Essentially they added to the original SCP additional constraints (of the same type as shown in equation (21) in order to increase the value of the linear programming relaxation of the (enlarged) SCP so that (eventually) it has the same value as the optimal SCP (integer) solution. Beasley (1990) presented a Lagrangian heuristic algorithm and reported that this heuristic gave better quality results than a number of other heuristics. Jacobs and Brusco (1993) developed a heuristic based on simulated annealing. Sen (1993) investigated the performances of a simulated annealing algorithm and a simple genetic algorithm on the minimal cost set covering problem. A comparative study of several different approximation algorithms for SCP was conducted in Grossman and Wool (1995).

Christofides and Korman (1975) presented a computational survey of a number of different methods for the SCP. Etcheberry (1977) presented an algorithm based upon Lagrangean relaxation of a subset of the covering constraints with subgadient optimization being used to determine the Lagrange multipliers. Paixao (1984) presented an algorithm based upon decomposition and state space relaxation. Beasley and Chu (1996) presented a genetic algorithm for the SCP. They proposed several modifications to the basic genetic procedure including a new fitness based crossover operator, a variable mutation rate and a heuristic feasibility operator tailored specifically to the SCP. Several neural network based algorithms were suggested or developed for problems related to the SCP [cf. Croall and Mason (1991), Jefries (1991)].

For the solution of problem (19) we apply the IP method with elements of a greedy algorithm and a genetic algorithm. We assume that the classification rule must correctly describe *most of the examples*, at least $A_{learning}$, by assumption. The measure of classification accuracy $A_{learning}$ is the ratio of examples correctly classified to the total number of examples, in percentage.

Suppose that we have a finite set of examples *U*, a finite set of attributes $A = \{a_1,...,a_K\} \cup \{a_d\}$. Let us class $Y_{v_{d,l}}$, for $v_{d,l} \in V_{a_d}$, and a set of positive examples S_P [cf. (3)], and a set of negative examples S_N [cf. (4)]. The consecutive steps of the algorithm are as follows:

Step 1. Set the initial values: $S = S_P$, i.e. the whole set of examples is initially assumed to contain the positive ones, S_N is a set of negative examples, and $R_W^* = \varnothing$, i.e. the initial set of complexes is assumed empty, iteration $r = 0$, given parameter $rel \geq 0$.

Step 2. Iteration $r = r + 1$. Determine the weights G by analyzing (pre-processing) of the examples due to (8).

Step 3. Determine an appropriate starting point; a good starting point may be a so-called centroid [cf. Kacprzyk and Szkatuła (1996)] that is some (possibly none existing) example in which the attributes take on values that occur most often in the positive examples and seldom in the negative examples.

The positive examples $e^m \in S_P$, $m = 1, ..., P$, are written as

$$e^m = [a_1 = v_{1,t(1,m)}; g_1(v_{1,t(1,m)})] \wedge ... \wedge [a_K = v_{K,t(K,m)}; g_K(v_{K,t(K,m)})] \tag{24}$$

while the negative examples $e^{P+n} \in S_N$, $n = 1, ..., N$ (for simplicity the upper index P is omitted below and later on when it does not lead to confusion), are written as:

$$e^n = [a_1 = v_{1,t(1,n)}; g_1(v_{1,t(1,n)})] \wedge ... \wedge [a_K = v_{K,t(K,n)}; g_K(v_{K,t(K,n)})] \tag{25}$$

For each attribute a_j, $j = 1, ..., K$, we determine such a value $v_{j,t(j,*)} \in V_{a_j}$ that [cf. (8)]

$$\max_{v_{j,l} \in V_{a_j}} \{ g_j(v_{j,l}) \} = \max_{v_{j,l} \in V_{a_j}} \left\{ \frac{1}{P} \sum_{m=1}^{P} \delta(e^m, v_{j,l}) - \frac{1}{N} \sum_{n=1}^{N} \delta(e^n, v_{j,l}) \right\} \tag{26}$$

where:

$v_{j,l} \in V_{a_j}$, $l = 1, ..., L_j$, V_{a_j} is the set of possible values of attribute a_j,

$$\delta(e^i, v_{j,l}) = \begin{cases} 1 & for \ v_{j,t(j,i)} = v_{j,l} \\ 0 & otherwise \end{cases}.$$

Now, we form an example, called a centroid

$$e^* = [a_1 = v_{1,t(1,*)}; g_1(v_{1,t(1,*)})] \wedge ... \wedge [a_K = v_{K,t(K,*)}; g_K(v_{K,t(K,*)})] \tag{27}$$

that contains the selectors with the most typical positive values of the particular attributes. Needless to say that this example may be artificial, i.e. nonexistent. The concept of a centroid is crucial for the efficiency of the algorithm.

We introduce a similarity measure η yielding a degree of similarity (from [0,1]) between the centroid e^* and a positive example e^m [cf. (24)] as, e.g.:

$$\eta(e^m, e^*) = \frac{1}{K} \sum_{j=1}^{K} \delta(e^m, v_{j,t(j,*)}) \tag{28}$$

or

$$\eta(e^m, e^*) = \sum_{j=1}^{K} g_j(v_{j,t(j,*)}) \cdot \delta(e^m, v_{j,t(j,*)}) \tag{29}$$

where $g_j(v_{j,t(j,*)})$ is the weight of the value $v_{j,t(j,*)}$ of attribute a_j obtained by using, e.g., (8).

Finally, in the set of positive examples we find such an example $e^p \in S_P$ for which

$$\max_{e^m \in S_P} \eta(e^m, e^*) \tag{30}$$

i.e. the closest positive example to e^*.

Then, as the starting point for the next iterations there is adopted a real (existing) example that is the closest, e.g., in the sense of the measure of similarity (28) or (29), to the one found by (30). This example e^p

$$e^p = [a_1 = v_{1,t(1,p)}; g_1(v_{1,t(1,p)})] \wedge ... \wedge [a_K = v_{K,t(K,p)}; g_K(v_{K,t(K,p)})]$$

is used in the algorithm proposed in the next steps as the starting point.

Step 4. For the e^p we form the matrix Z_{N*K}=$[z_{nj}]$, $n = 1, ..., N$, $j = 1, ..., K$, due to (16) and a modification of the set covering problem such that

$$\min_{x,\gamma} \sum_{j=1}^{K} c_j x_j$$

with linear constrains

$$\sum_{j=1}^{K} z_{nj} x_j \geq \gamma_n, \qquad n = 1, ..., N$$

$$\sum_{n=1}^{N} \gamma_n \geq N - rel$$

where:

$c_j = (1 - g_j(v_{j,t(j,p)}))$, $z_{nj} \in \{0,1\}$, $x_j \in \{0,1\}$, $n = 1, ..., N$, $j = 1, ..., K$,
$\gamma = [\gamma_1, ..., \gamma_N]^T$ is a 0-1 vector, $rel \geq 0$.

Step 5. To solve above problem we apply a greedy algorithm IP_GRE or genetic algorithm IP_GA. The 0-1 vector $x^* = [x_1^*, ..., x_K^*]^T$ found in arbitrary way determines in a unique way the complex $C_W^{l,*}$ and the 0-1 vector $\gamma = [\gamma_1, ..., \gamma_N]^T$ determines the fulfilled constrains.

Step 6. Include complex $C_W^{l_r\,*}$ found in Step 5 into the classification rule sought R_W^* (i.e. that with the minimal weighted length), $R_W^* := R_W^* \cup C_W^{l_r\,*};q(C_W^{l_r\,*})$, where $q(C_W^{l_r\,*})$ is the strength of the r-th elementary rule, and discard from the set of positive examples S all examples covered by complex $C_W^{l_r\,*}$.

Step 7. If the set of examples S remaining is *small enough*, STOP and the rule

$$R_W^* = C_W^{l_1\,*};q(C_W^{l_1\,*}) \cup \ldots \cup C_W^{l_L\,*};q(C_W^{l_L\,*}) \tag{31}$$

$l_1, \ldots, l_L \subseteq \{1, \ldots, K\}$, is the one sought; otherwise, return to Step 2.

Example 2. Assume that we have two class $Y_{v_{d,l_1}}$, $Y_{v_{d,l_2}}$ for $v_{d,l_1} \in V_{a_d}$, $v_{d,l_2} \in V_{a_d}$. Suppose that in Fig. 3 all the training examples are shown. Those belonging to class $Y_{v_{d,l_1}}$ are marked by $\oplus$ (11 examples) and those belonging to class $Y_{v_{d,l_2}}$ are marked by Θ (11 examples).

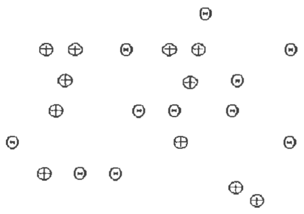


Figure 3.

Let consider the class $Y_{v_{d,l_1}}$. First, we find the set of elementary rules for the class $Y_{v_{d,l_1}}$ that correctly describe the most training examples, for *rel* = 1, i.e. $C^{l_1} \cup \ldots \cup C^{l_4} \to [\,a_d = v_{d,l_1}\,]$, is illustrated in Fig. 4.

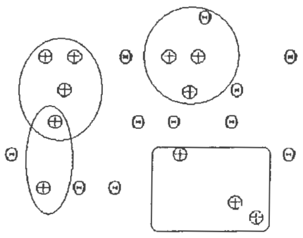


Figure 4.

Next, we determine in the same way the set of rules for the class $Y_{v_{d,l_2}}$, i.e. $C^{l_1} \cup \ldots \cup C^{l_5} \to [\,a_d = v_{d,l_2}\,]$, is illustrated in Fig. 5.

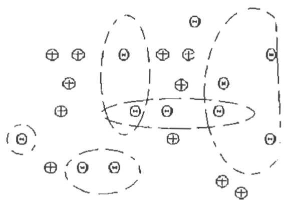


Figure 5.

The rules formed in this manner can be applied to classification of new examples, i.e. ones that have not appeared in the learning process. Such a classification is carried out through verification of fulfilment of conditions in the conditional parts of the rules, and in case of equivocal situations (when more than one, or none, of the rules is fulfilled), the degree of matching of the class is accounted [cf. Szkatuła, 1995].

4. The IP_GRE Procedure

To solve problem (20), (21), (22), (23) (i.e. Step 5 in the IP procedure) we apply a modification of the greedy algorithm IP_GRE. The example e^p, found in Step 3 of the IP procedure, is used in the algorithm as the starting point. For the e^p we form the matrix $Z_{N*K}=[z_{nj}]$, $n=1,\dots,N$, $j=1,\dots,K$, due to (16). We assume initially that $x^*=[x_1^*,\dots,x_K^*]^T=[0,\dots,0]^T$ and $\gamma^*=[\gamma_1^*,\dots,\gamma_N^*]=[0,\dots,0]^T$, $rel\geq 0$.

Step 1'. We assume the initial matrix to be $M := Z$.

Step 2'. We calculate the efficiency of element x_j, $j=1,\dots,K$ with respect to the matrix M which is defined as

$$E(x_j,M)=\sum_{n=1}^{N} m_{nj}/(1-g_j(v_{j,t(j,p)})) \tag{32}$$

We choose the highest value $E(x_{j^*},M)$, and set $x_{j*}^*=1$.

Step 3'. We denote the j*-th column of M as $m_{j^*}=[m_{1j^*},m_{2j^*},\dots,m_{Nj^*}]^T$ and calculate new matrix M as

$$((M)^T\times diag(1-m_{1j^*},1-m_{2j^*},\dots,1-m_{Nj^*}))^T \tag{33}$$

and we obtain new vector $\gamma^*=[\gamma_1^*,\dots,\gamma_N^*]$ as

$$[\max\{m_{1j^*},\gamma_1^*\},\dots,\max\{m_{Nj^*},\gamma_N^*\}]^T \tag{34}$$

Step 4'. If $\sum_{n=1}^{N} \gamma_n^* \geq N - rel$ then STOP, otherwise return to Step 2'.

The 0-1 vector $x^* = [x_1^*, ..., x_K^*]^T$ found in such a way determines in a unique way the complex C_W^* sought and the 0-1 vector $\gamma^* = [\gamma_1^*, ..., \gamma_N^*]$ determines the fulfilled constrains. Now, we can go to Step 6 of the IP algorithm.

5. The IP_GA Procedure

For the solution of problem (20), (21), (22), (23) (i.e. Step 5 IP procedure) we apply modification of genetic algorithm. A genetic algorithm (GA) can be understood as an intelligent probabilistic search algorithm which can be applied to a variety of combinatorial optimization problems. It requires a set of individual elements (i.e. population) to be initialized. Each individual in the population is encoded into a string that represents a possible solution to a given problem. The fitness of a solution is evaluated with respect to a given objective function. Highly fit solutions are given opportunities to reproduce by exchanging pieces of their bits in the strings, in crossover procedure, with other highly fit solutions. Mutation is often applied after crossover by altering some bits in the strings. This produces new solutions that can either replace the whole population or replace less fit solutions. This evaluation-selection-reproduction cycle is repeated until a satisfactory solution is found. The theoretical foundations of genetic algorithms were originally developed by Holland (1995).

The basic steps of a simple genetic algorithm are shown below:

```
Set $t$ = 1. Generate an initial population $P(t)$ of possible solutions;
    Evaluate fitness of solutions in the population;
while a satisfactory solution has been found do
begin
    Select solutions from the population;
    Recombine solutions to produce new solutions;
    Evaluate fitness of new solutions;
    Set $t = t+1$. Replace some or all of the solutions in the population by the new
    solutions and create the new population P($t$).
end;
```

The first step in designing a genetic algorithm for a particular problem is to devise a suitable representation scheme, i.e. a way to represent a possible solution in a population. We assume a *K*-bit binary string which represents the potential solution structure, where *K* is the number of variables in our problem (i.e. columns in the SCP). In this representation a value 1 for the *j*-th bit implies that column *j* is in the solution x^l, i.e. that x_j^l is in the solution, respectively. This binary representation of the solution $x^l = [x_1^l, x_3^l, x_4^l, ..., x_{K-1}^l]^T$ as $[1,0,1,...,1,0]^T$ is illustrated in Figure 6.

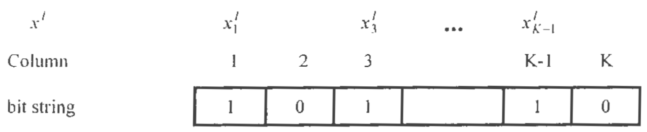


Figure 6. Binary representation of solution x^l as $[1,0,1,\ldots,1,0]^T$

We note that a bit string might represent an infeasible solution. An infeasible solution is one for which at least one of constraints is violated, i.e. $\sum_{n=1}^{K} z_{nj}x_j = 0$ for some $n \in I$, where $I = \{1,\ldots,N\}$ is the set of rows. There are a number of standard ways of dealing with constrains and infeasible solutions in genetic algorithms:

- to use a special representation that automatically ensures that all solutions are feasible,
- to separate the evaluation of fitness and infeasibility [Chu and Beasley (1995)],
- to design a heuristic repair operator which guarantees to transform any infeasible solution into a feasible solution [Beasley and Chu (1996)],
- to apply a penalty function [Goldberg (1989), Smith and Tate (1993)] to penalize the fitness of any infeasible solution without distorting the fitness landscape.

In our procedure, IP_GA, in each iteration all solutions are evaluated with respect to their completeness and consistency. We adopted a simple approach of using a penalty/evaluation function which assigns utility to candidate solutions. The fitness of an individual solution *x* is calculated simply by

$$eval(x) = f(x) - g_{\max}\frac{K}{N}\sum_{n=1}^{N} f_n(x) \tag{35}$$

where:

$$f(x) = \sum_{j=1}^{K} c_j x_j$$

$$f_n(x) = \begin{cases} 0 & for \ \sum_{j=1}^{K} z_{nj}\cdot x_j > 0 \\ 1 & for \ \sum_{j=1}^{K} z_{nj}\cdot x_j = 0 \end{cases}$$

$$g_{\max} = \max\{g_j : j = 1,\ldots,K\}, \ n=1,\ldots,N$$

where x_j is the value of the *j*-th column in the string corresponding to the solution x and c_j is the cost of *j*-th column.

The second step in designing a genetic algorithm is to generate an initial population of feasible solutions. In our problem the initial population was randomly generated and the size of the population was proportional to the number of columns.

The third step is a particular choice of crossover and mutation operators which are applied to the new population.

We arbitrarily adopted the crossover operator which for two solutions forms two new solutions. Under the one point crossover operator, two structures in the population exchange portions of their binary representations. This can be implemented by choosing a point at random, called a crossover point, and exchanging the segments to the right of this point. For example, select two solutions x^1 and x^2 from the population

$$\begin{array}{l|l} x^1 = [x_1^1, x_2^1, & x_3^1, \ldots, x_{K-1}^1, x_K^1]^T \\ x^2 = [x_1^2, x_2^2, & x_3^2, \ldots, x_{K-1}^2, x_K^2]^T \end{array}$$

and suppose that the crossover one point has been chosen as indicated. Then, we obtain two new structures:

$$\begin{array}{l|l} x^{1'} = [x_1^1, x_2^1, & x_3^2, \ldots, x_{K-1}^2, x_K^2]^T \\ x^{2'} = [x_2^2, x_2^2, & x_3^1, \ldots, x_{K-1}^1, x_K^1]^T . \end{array}$$

After the new population has been selected, mutation is applied to each structure in the new population. The mutation procedure is performed that mutates some randomly selected bits in the solution. It works by inverting each bit in the solution with some probability (the mutation rate). The rate of mutation is generally set to be a small value. For example, by selecting a solution x^l from the population

$$x^l = [x_1^l, x_2^l, x_3^l, \ldots, x_{K-1}^l, x_K^l]^T$$

we obtain

$$x^{l'} = [x_1^l, \bar{x}_2^l, x_3^l, \ldots, x_{K-1}^l, x_K^l]^T .$$

The fourth step is the choice of a selection method. Selection is the process of choosing structures for the next generation from the structures in the current generation. The structures of new population are chosen by a stochastic universal sampling Baker (1987). This method uses a single wheel spin. This wheel is spun with a number of equally spaced markers equal to the population size. The selection pointers are then randomly shuffled and the selected structures are copied into the new population. A new population is formed with those better solutions more likely to appear and the cycle repeats.

Our procedure steps are as follows:

Step 1'. Set $t = 1$. Generate an initial population of random solutions $P(t) = \{x^1, x^2, ..., x^P\}$. Each solution is simply a binary string of length *K*. Evaluate the fitness $eval(x^l)$ of individuals in the population, $l = 1,2, ..., P$.

Step 2'. A mutation operator is applied to each solution in the population.

Step 3'. For the first solutions (the crossover rate multiplied by the size of the population) a crossover operator is applied. Two solutions are chosen and form two new solutions.

Step 4'. The new solution generated by the crossover and mutation procedures may not be feasible because the constraints may not all be satisfied. We evaluate the fitness $eval(x^l)$ of new individuals in the population.

Step 5'. If a termination condition satisfied STOP, the best solution found is the one with the smallest fitness in the population; otherwise, go to Step 6'.

Step 6'. Select a new population *P(t+1)* from population *P(t)* and return to Step 2'

The 0-1 vector $x^* = [x_1^*, ..., x_K^*]^T$ found by this way determines in a unique way the complex C_W^* sought. Now, we can go to Step 6 of IP algorithm.

The IP_GA algorithm described above is relatively simple and efficient. It requires a number of parameters, e.g. the population size, probabilities of applying genetic operators, etc.

6. Application of the IP Algorithm to Solve Test Problems

6.1. Solution of Test Example

Suppose that the examples are described by the following three attributes (concerning some features of human beings):

a_1: "height" with values from the set {"high", "low"},
a_2: "color_of_hair" with values from the set {"dark", "red", "blond"},
a_3: "color_of_eyes" with values from the set {"blue", "green"}

and a decision attribute "*class*" with values from the set {"class 1", "class 2"}. Suppose that we have three examples belonging to class 1:

e^1: [height = low] [color_of_hair = blond] [color_of_eyes = blue]
e^2: [height = high] [color_of_hair = red] [color_of_eyes = green]
e^3: [height = high] [color_of_hair = blond] [color_of_eyes = blue]

and five examples belonging to class 2:

e^4 : [height = high] [color_of_hair = blond] [color_of_eyes = green]
e^5 : [height = low] [color_of_hair = dark] [color_of_eyes = blue]
e^6 : [height = high] [color_of_hair = dark] [color_of_eyes = blue]
e^7 : [height = high] [color_of_hair = dark] [color_of_eyes = green]
e^8 : [height = low] [color_of_hair = blond] [color_of_eyes = green]

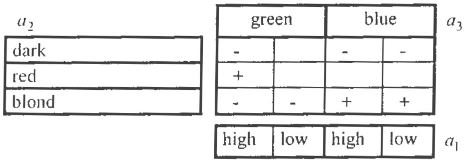
notice that both types are written as in (6) and (7) but with the "$\wedge$" omitted for simplicity.

We wish to find the two classification rules:

$$R_{IV}^{1*} \rightarrow [\text{class} = \text{class } 1],\ R_{IV}^{2*} \rightarrow [\text{class} = \text{class } 2]$$

where R_{IV}^{1*} and R_{IV}^{2*} are disjunction of the complexes to be found.

To determine the first classification rule, R_{IV}^{1*}, we will use the algorithm IP proposed in the previous sections. To visualize the results we use two-dimensional diagrams. Each box of the diagram represents the conjunction of some values of the attribute a_2 (rows) and a_1, a_3 (columns). In Fig. 7 all the training examples are shown. Those belonging to class 1 (positive examples, S_P) are marked by "+", and those belonging to class 2 (the negative examples, S_N) are marked by "-".



a_2 \ a_3	green		blue	
dark	-		-	-
red	+			
blond	-	-	+	+
a_1	high	low	high	low

Figure 7. Training examples: the positive examples are marked by "+", the negative examples are marked by "-"

The steps of the IP algorithm are as follows.

Step 1. We denote the two disjoint sets of examples as:

$$S_P = \{ e^1, e^2, e^3 \},$$
$$S_N = \{ e^4, e^5, e^6, e^7, e^8 \},$$

$R_{IV}^{1*} = \varnothing$, $rel = 1$ (i.e. the classification rule may don't describe no more than one training example), and assume the initial set to be $S = S_P$.

Step 2. Determine the weights G by analyzing (pre-processing) of the examples due to (8). For each of the value of attribute a_j, $j = 1,2,3$, we calculate

$$g_j(v) = \frac{1}{3}\sum_{m=1}^{3}\delta(e^m, v) - \frac{1}{5}\sum_{n=4}^{8}\delta(e^n, v) \quad \text{for each } v \in V_{a_j},\ j = 1,2,3.$$

We obtain, therefore, first, for the attribute a_1:"height":

$$g_1(low) = \frac{1}{3}\sum_{m=1}^{3}\delta(e^m, low) - \frac{1}{5}\sum_{n=4}^{8}\delta(e^n, low) = 1/3 - 2/5 = -1/15$$

$$g_1(high) = \frac{1}{3}\sum_{m=1}^{3}\delta(e^m, high) - \frac{1}{5}\sum_{n=4}^{8}\delta(e^n, high) = 2/3 - 3/5 = 1/15$$

Then, for the second attribute, a_2:"color_of_hair", we obtain

$$g_2(blond) = \frac{1}{3}\sum_{m=1}^{3}\delta(e^m, blond) - \frac{1}{5}\sum_{n=4}^{8}\delta(e^n, blond) = 2/3 - 2/5 = 4/15$$

$$g_2(red) = \frac{1}{3}\sum_{m=1}^{3}\delta(e^m, red) - \frac{1}{5}\sum_{n=4}^{8}\delta(e^n, red) = 1/3$$

$$g_2(dark) = \frac{1}{3}\sum_{m=1}^{3}\delta(e^m, dark) - \frac{1}{5}\sum_{n=4}^{8}\delta(e^n, dark) = -3/5.$$

Finally, for the third attribute, a_3:"color_of_eyes", we obtain:

$$g_3(blue) = \frac{1}{3}\sum_{m=1}^{3}\delta(e^m, blue) - \frac{1}{5}\sum_{n=4}^{8}\delta(e^n, blue) = 2/3 - 2/5 = 4/15$$

$$g_3(green) = \frac{1}{3}\sum_{m=1}^{3}\delta(e^m, green) - \frac{1}{5}\sum_{n=4}^{8}\delta(e^n, green) = 1/3 - 3/5 = -4/15.$$

Step 3. For each of the three attributes we calculate, using (26), the value of v_j^*, $j = 1,2,3$, i.e. we calculate subsequently for all values of the particular attribute the value of

$$v_j^* = \arg \max_{v \in V_{a_j}} \left\{ \frac{1}{3}\sum_{m=1}^{3}\delta(e^m, v) - \frac{1}{5}\sum_{n=4}^{8}\delta(e^n, v) \right\} = \arg \max_{v \in V_{a_j}} \{ g_j(v) \}.$$

For the attribute "height" we obtain:

$$v_1^* = \arg\ \max\{g_1(low), g_1(high)\} = \text{high}.$$

For the second attribute, "color_of_hair", we obtain:

$$v_2^* = \arg\ \max\{g_2(blond), g_2(red), g_2(dark)\} = \text{red}.$$

Finally, for the third attribute, "color_of_eyes", we obtain:

$$v_3^* = \arg\ \max\{g_3(blue), g_3(green)\} = \text{blue}.$$

The centroid, e^* (see Fig. 8), is therefore e^* = [height = high][color_of_hair = red] [color_of_eyes = blue].

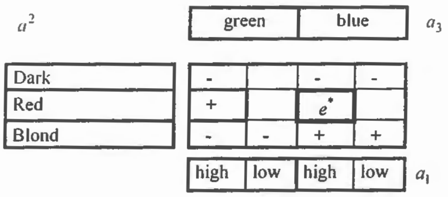


Figure 8. Centroid e^* and the starting point e^2 for the algorithm that is marked by shaded box

Calculating for all the positive examples e^m from S the value of $\eta(e^m, e^*)$ using (29), we obtain:

$$\eta(e^1, e^*) = 4/15,$$
$$\eta(e^2, e^*) = 1/15 + 1/3 = 2/5,$$
$$\eta(e^3, e^*) = 1/15 + 4/15 = 1/3$$

and hence, due to (30) we take the example e^2 as the starting point for the algorithm, see Fig. 9.

Step 4. For the e^2 mentioned above, i.e.

e^2: [height = high][color_of_hair = red] [color_of_eyes = green]

and all the negative examples

e^4: [height = high][color_of_hair = blond] [color_of_eyes = green]
e^5: [height = low][color_of_hair = dark] [color_of_eyes = blue]
e^6: [height = high][color_of_hair = dark] [color_of_eyes = blue]
e^7: [height = high][color_of_hair = dark] [color_of_eyes = green]
e^8: [height = low][color_of_hair = blond] [color_of_eyes = green]

we form the matrix $Z_{N*K}=[z_{nj}]$ n=1,2,3,4,5, j=1,2,3 due to (16):

$$Z = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

Problem (18) can be written as a modification of the set covering problem (SCP)

$$\min_{x,\gamma} \sum_{j=1}^{3} c_j x_j$$

subject to

$$\sum_{j=1}^{3} z_{nj} x_j \geq \gamma_n \qquad n = 1, \ldots, 5$$

and additional constraint

$$\sum_{n=1}^{5} \gamma_n \geq 4$$

where:

$c_j = (1 - g_j(v_j))$, $z_{nj} \in \{0,1\}$, $x_j \in \{0,1\}$, $j = 1,2,3$, $n = 1,2,3,4,5$,
$\gamma = [\gamma_1, \ldots, \gamma_5]^T$ is a 0-1 vector.

Step 5. Step 5 of algorithm may be solved by a greedy or a genetic algorithm. The algorithm was implemented as the IP algorithm with the IP_GRE and IP_GA procedure. To solve problem we apply the greedy algorithm IP_GRE. We assume initially $x^* = [x_1^*, x_2^*, x_3^*]^T = [0,0,0]^T$

Step 1'. We assume initially *M:=Z* and $\gamma^* = [\gamma_1^*, \ldots, \gamma_5^*] = [0, \ldots, 0]^T$, ***rel*** = 1, i.e. at most one examples may not be correctly covered, by assumption.

Step 2'. We calculate using (32) the efficiencies of the elements x_1, x_2, x_3 with respect to the matrix *M:*

$E(x_1, M) = 15/14 \times 2 = 15/7,$
$E(x_2, M) = 3/2 \times 5 = 15/2,$
$E(x_3, M) = 15/19 \times 2 = 30/19.$

We choose the highest value, and set $x_2^* := 1$, i.e. $x^* = [0,1,0]^T$.

Step 3'. We obtain new vector $[\gamma_1^*, \ldots, \gamma_5^*]^T$ as $[1, \ldots, 1]^T$. Vector x^* don't describe all the negative examples.

Step 4'. Since $\sum_{n=1}^{5} \gamma_n^* = 5 \geq 4$ than STOP and go to Step 5.

We obtain $x^* = [0,1,0]^T$ and $[\gamma_1^*, \ldots, \gamma_5^*]^T = [1,1,1,1,1]^T$, its related optimal complex as

C_W^{1*} = [color_of_hair = red], $\quad q(C_W^{1*}) = 1 \backslash 8 = 0.125$.

In Fig. 10 complex C_W^{1*} is shown as left-shaded boxes.

Step 6. We "add" (via "$\cup$") the complex C_W^{1*} found in Step 5 to the classification rule sought R_W^{1*}, $R_W^{1*} = R_W^{1*} \cup C_W^{1*}$; 0.125. From the set of positive examples S we remove all examples described by that complex C_W^{1*}; i.e. we remove e^2, and hence $S = \{e^1, e^3\}$, see Fig. 9.

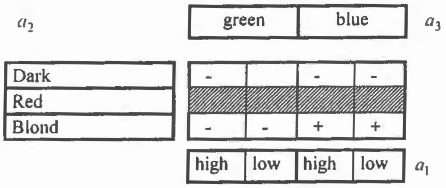


Figure 9. Complex C_W^{1*} and the training examples for the next iteration

Step 7. Since S is not small enough, then we repeat Step 2.

Step 2. Determine the weights G by analyzing (pre-processing) of the examples due to (8). For each of the value of attributes we calculate:

$$g_j(v) = \frac{1}{2}\sum_{m=1,3} \delta(e^m, v) - \frac{1}{5}\sum_{n=4}^{8} \delta(e^n, v)$$

for each $v \in V_{a_j}$, $j = 1,2,3$. We obtain therefore, first, for the attribute "height":

$$g_1(low) = \frac{1}{2}\sum_{m=1,3} \delta(e^m, low) - \frac{1}{5}\sum_{n=4}^{8} \delta(e^n, low) = 1/2 - 2/5 = 1/10$$

$$g_1(high) = \frac{1}{2}\sum_{m=1,3} \delta(e^m, high) - \frac{1}{5}\sum_{n=4}^{8} \delta(e^n, high) = 1/2 - 3/5 = -1/10.$$

Then, for the second attribute, "color_of_hair", we obtain:

$$g_2(blond) = \frac{1}{2}\sum_{m=1,3} \delta(e^m, blond) - \frac{1}{5}\sum_{n=4}^{8} \delta(e^n, blond) = 1 - 2/5 = 3/5$$

$$g_2(red) = \frac{1}{2}\sum_{m=1,3} \delta(e^m, red) - \frac{1}{5}\sum_{n=4}^{8} \delta(e^n, red) = 0$$

$$g_2(dark) = \frac{1}{2}\sum_{m=1,3} \delta(e^m, dark) - \frac{1}{5}\sum_{n=4}^{8} \delta(e^n, dark) = -3/5$$

Finally, for the third attribute, "color_of_eyes", we obtain:

$$g_3(blue) = \frac{1}{2}\sum_{m=1,3}\delta(e^m, blue) - \frac{1}{5}\sum_{n=4}^{8}\delta(e^n, blue) = 1 - 2/5 = 3/5$$

$$g_3(green) = \frac{1}{2}\sum_{m=1,3}\delta(e^m, green) - \frac{1}{5}\sum_{n=4}^{8}\delta(e^n, green) = -3/5.$$

Step 3. For each of the three attributes we calculate, using (26), the value of v_j^*, $j = 1,2,3$, i.e. we calculate subsequently for all values of the particular attribute a_j the value of

$$v_j^* = \arg\max_{v \in V_{a_j}} \left\{ \frac{1}{2}\sum_{m=1,3}\delta(e^m, v) - \frac{1}{5}\sum_{n=4}^{8}\delta(e^n, v) \right\}$$

For the attribute "height" we obtain:

$$v_1^* = \arg\max\{g_1(low), g_1(high)\} = \text{low}.$$

For the second attribute, "color_of_hair", we obtain:

$$v_2^* = \arg\max\{g_2(blond), g_2(red), g_2(dark)\} = \text{blond}.$$

Finally, for the third attribute, "color_of_eyes", we obtain:

$$v_3^* = \arg\max\{g_3(blue), g_3(green)\} = \text{blue}.$$

The centroid is therefore e^* = [height = low][color_of_hair = blond] [color_of_eyes = blue] and in this case it is one of the existing examples (e^1 from the set of positive examples) though it need not always be the case. Usually, the centroid does not constitute any of the positive examples, and for each positive example one should calculate the value of v_2^*, e.g., due to (26). In Fig. 10 centroid e^* which is equal existing examples e^1 is marked by shaded box .

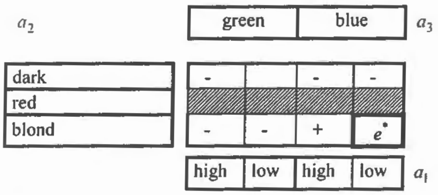


Figure 10. Centroid e^* is equal existing examples e^1 which is marked by shaded box

We take the example e^1 as the starting point for the algorithm. We assume initially $x^* = [x_1^*, x_2^*, x_3^*]^T = [0,0,0]^T$ and $\gamma^* = [\gamma_1^*, ..., \gamma_5^*]^T = [0, ...,0]^T$.

Step 4. For the example e^1, i.e. e^1 = [height = low][color_of_hair = blond] [color_of_eyes = blue] and all the negative examples:

e^4: [height = high] [color_of_hair = blond] [color_of_eyes = green]
e^5: [height = low] [color_of_hair = dark] [color_of_eyes = blue]
e^6: [height = high] [color_of_hair = dark] [color_of_eyes = blue]
e^7: [height = high] [color_of_hair = dark] [color_of_eyes = green]
e^8: [height = low] [color_of_hair = blond] [color_of_eyes = green]

we form the matrix Z due to (16):

$$Z = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Step 5. To solve the problem in Step 4 we apply the greedy algorithm IP_GRE.

Step 1'. We assume initially $M:=Z$ and $[\gamma_1^*, ..., \gamma_5^*]^T = [0, ..., 0]^T$, $rel = 1$.

Step 2'. We calculate using (33) the efficiencies of the elements x_1, x_2, x_3 with respect to the matrix M:

$E(x_1, M) = 10/9 \times 3 = 10/3$
$E(x_2, M) = 5/2 \times 3 = 15/2$
$E(x_3, M) = 5/2 \times 3 = 15/2.$

We choose the highest value, and set $x_2^* := 1$, i.e. $x^* = [0,1,0]^T$.

Step 3'. We calculate new matrix M:

$$M = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and a new vector $\gamma^* = [\gamma_1^*, ..., \gamma_5^*]^T = [0,1,1,1,0]^T$. Vector x^* does not describe two negative examples.

Step 4'. Since $\sum_{n=1}^{5} \gamma_n^* = 3 < 4$ then we return to *Step 2'*.

Step 2'. We calculate using (32) the efficiencies of the elements x_1, x_2, x_3 with respect to the matrix M:

$E(x_1, M) = 10/9 \times 1 = 10/9$
$E(x_2, M) = 0$
$E(x_3, M) = 5/2 \times 2 = 5.$

We choose the highest value, and set $x_3^* := 1$, i.e. $x^* = [0, 1, 1]^T$.

Step 3'. We calculate new vector $\gamma^* = [\gamma_1^*, \dots, \gamma_5^*]^T = [1, \dots, 1]^T$. Vector x^* does not describe all the negative examples.

Step 4'. Since $\sum_{n=1}^{5} \gamma_n^* = 5 \geq 4$ than STOP and go to Step 5.

The vector $x^* = [0, 1, 1]^T$ found by greedy algorithm determines in a unique way the complex C_W^{2*} sought as

C_W^{2*} = [color_of_hair = blond] [color_of_eyes = blue],
$q(C_W^{2*}) = 2/8 = 0.25$.

In Fig. 11 complex C_W^{2*} is shown by right-shaded boxes.

Step 6. We add the complex C_W^{2*} found in Step 5 to the rule sought R_W^{1*}, $R_W^{1*} = R_W^{1*} \cup C_W^{2*}$; 0.250. From the set of positive examples S delete all examples covered by this complex C_W^*, i.e. we delete e^1 and e^2.

Step 7. The set of positive examples S is small enough, is empty. The classification rule sought is therefore

$R_W^{1*} = C_W^{1*}; 0.125 \cup C_W^{2*}; 0.250$, i.e.

[color_of_hair = red];0.125 $\cup$
[color_of_hair=blond] $\wedge$ [color_of_eyes_blue]}; 0.250
$\rightarrow$ [class = class 1]

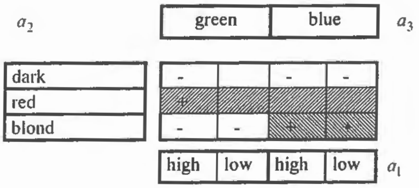


Figure 11. Complex C_W^{1*} (marked by left-shaded boxes) and C_W^{2*} (marked by right-shaded boxes)

6.2. Solving a Monk's Problem

Let consider a standard example known as the Monk's (M3) problem described by seven discrete-valued attributes:

a_1: head shape	{round, square, octagon},
a_2: body shape	{round, square, octagon},
a_3: is smiling	{yes, no},
a_4: holding	{sword, balloon, flag},
a_5: jacket color	{red, yellow, green, blue},
a_6: has tie	{yes, no},
a_7 : class	{class 1, class 0}.

The training data set includes 122 examples (with 6 misclassifications) which represented 30% of the total event space. The classification is binary. The testing date includes all possible examples (432 examples).

To visualize the results we use two-dimensional diagram [c.f. Wnek, Sarma, Wahab, Michalski, 1991]. Basically, each part (box) of the diagram (which consists of 18 rows and 24 columns) represents the conjunction of some values of the attributes a_1, a_2, a_3 (rows) and a_4, a_5, a_6 (columns).

On the first diagram (Fig. 12) all the training examples are shown. Those belonging to class 1 (say, the positive examples) are marked by "+", and those belonging to class 0 (say, the negative examples) are marked by "-". Misclassifications are marked by the boxes □.

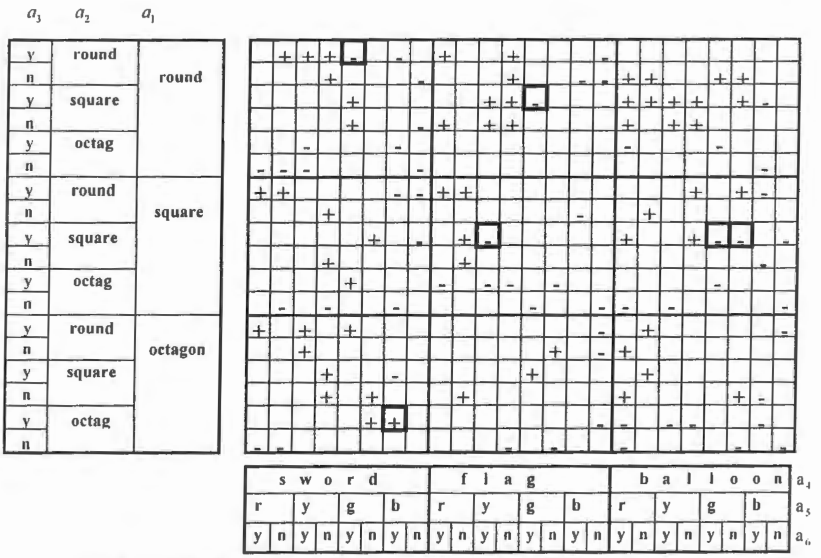


Figure 12. The training examples

On the second diagram have been presented all the training examples and the classification rules created by IP method (Fig. 13).

The rules derived are shown as follows:

Classification into class 1 is marked by left-shaded boxes	▧
Classification into class 0 is marked by right-shaded boxes	▨
Classification simultaneously into class 1 and 0 is marked by boxes	■
Example does not fulfill rules is marked by boxes	□

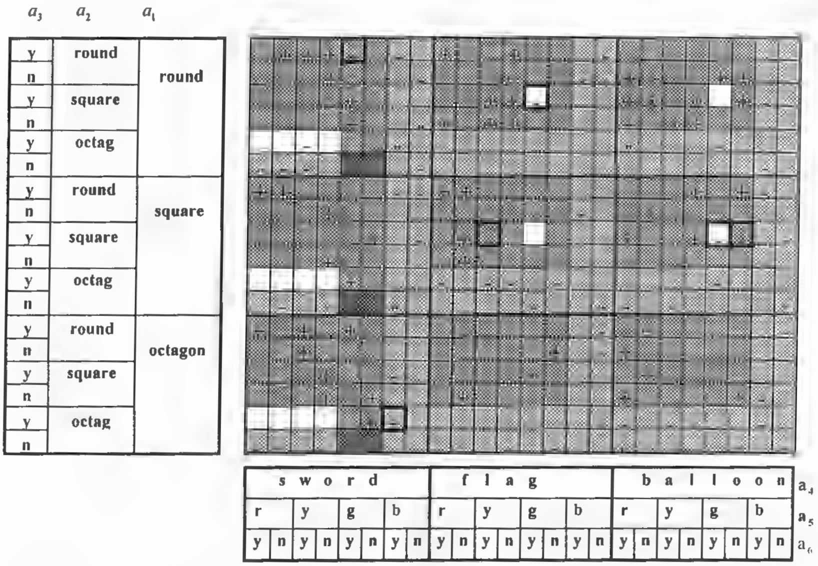


Figure 13. Training examples and created classification rules.

On the third diagram (Fig. 14) have been presented all testing examples and the results of the IP method. We will plot the results in the same way: "+" indicates that the algorithm classifies the entity as a member of the class 1, and "-" as a member of the class 0. Additional square will indicate misclassifications.

The classification accuracy of the rules derived is the percentage of testing examples correctly classified.

a_3	a_2	a_1	s w o r d								f l a g								b a l l o o n								a_4
			r		y		g		b		r		y		g		b		r		y		g		b		a_5
			y	n	y	n	y	n	y	n	y	n	y	n	y	n	y	n	y	n	y	n	y	n	y	n	a_6
y	round	round	+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	
n			+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	
y	square		+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	
n			+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	
y	octag		-	-	-	-	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
n			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
y	round	square	+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	
n			+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	
y	square		+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	
n			+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	
y	octag		-	-	-	-	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
n			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
y	round	octagon	+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	
n			+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	
y	square		+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	
n			+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	+	+	+	+	+	+	-	-	
y	octag		-	-	-	-	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
n			-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	

Figure 14. Testing examples and results of learning algorithm (accuracy 98.6%)

We will compare the results obtained by using IP method with the following ones from the literature:

- ID3, ID5R, AQR and CN2 – At the Departament of Komputer Scence, University of Karlsruhe, Germany [Kreuziger, Hamann, Wenzel 1991].
- PRISM – At the Institute of Informatics, University of Zurych, Switzerland [Keller 1991].
- AQ17-DCI and AQ17-FCLS – At the Artificial Intelligence Centra, Georgie Mason University [Bala, Bloedorn, Jong, Kaufman, Michalski, Pachowicz, Vafaie, Wnek, Zhang 1991].
- IP – the one proposed in this paper.

The results are presented in Fig. 15, and one can notice that the IP implementation attains the highest classification accuracy (ca. 98%) among the techniques considered.

One significant characteristic of this comparison is that the results are less biased than in comparisons performed by a single person advocating a specific learning method, and more accurately reflect the generalization behavior of the learning techniques as applied by knowledgeable users.

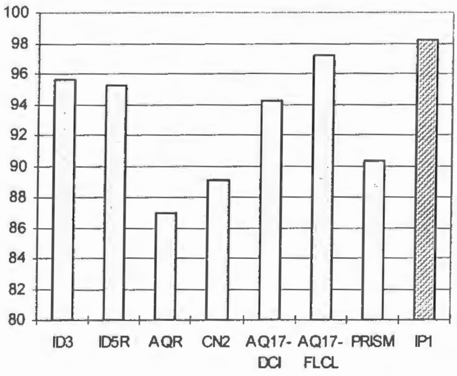


Figure 15. Classification accuracy of different methods machine learning from examples

7. Application of the IP Algorithm to Solve Some Medical Problems

7.1. Solving a Thyroid Cancer Problem

The medical data set published by Nakache and Asselian have been collected on patients with thyroid cancer at Hospital Ambroise Paré, from 1960 to 1980. They concern 281 patient, all submitted to a surgical treatment. The time of analysis has been fixed in July 1981. At that time, the patient is dead or alive. The survival time is then fully known for those patients who have died before the time of analysis. For those who are still alive, we only know the inferior limit of their survival time, cf. Fig. 16. Patients who survive may be completely cured and their survival pattern might have no relationship to those who die.

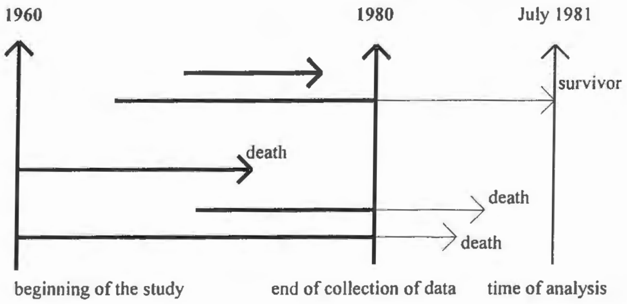


Figure 16. Retrospective medical data set

Each patient was described by the following 12 attributes in a discrete coding:

sex,	{male, female}
age,	{<40, 40-60, 60-70, >70}
histology,	{well differentiated, poorly differentiated}

metastasis,	{yes, no}
enlargement,	{uni-lobe, uni-lobe+isthm, all the thyroid}
clinical lymph nodes,	{yes, no}
clinical aspect,	{unique nodule, multi nodules, important enlargement}
pathological lymph nodes,	{yes, no}
compressive syndromes,	{yes, no}
invasion,	{no, small, average, large}
survival time,	{in months}, length in month of survival time from the entrance in the study (between 1960 and 1980) to the time of analysis,
survival,	{survivor, non survivor at time of analysis}.

Two of the 12 attributes are important: survival time (in month) of a patient at the time of analysis, and survival or non survival at the time of analysis.

Looking at each attribute separately, we had 229 survivors and 52 dead during the study. Among 52 patients there were 14 patients with missing values. The dependence between the time of survival and the number patients was presented in Fig. 17.

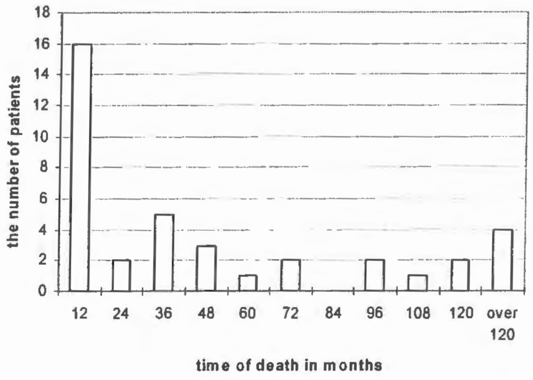


Figure 17. Dependence between the time of death in months and the number of patients

The aim of such a study is to identify prognostic elements of disease evolution and to define a prognostic rule for a new case coming from the same population and being in the same conditions.

The patients (training examples) have been divided into two classes. The class attribute is the survival time. The first class includes patients with the survival time over 7 years. The patients with the survival time below 7 years belong to the second class. Thus, 48 examples belong to the first and 29 examples belong to the second class. The problem of learning from examples is formulated as to find two classification rules:

$R_{W}^{1*} \to [class = class1]$, $R_{W}^{2*} \to [class = class2]$.

The R_W^{l*} is specified by "elementary" conditions depending on the attributes, $l=1,2$, for the classes:

class 1: the patients will be alive over 7 years,
class 2: the patient will dead during 7 years,

using all examples as patterns.

The IP method was applied to the data described above. The results of applying the method to medical data are presented and described below

We use IP_GRE (with elements of a greedy algorithm) and $A_{learning}$ = 100%. The measure of classification accuracy $A_{learning}$ is the ratio of examples correctly classified to the class 1 (or class 2) to the total number of examples, in percent. The classification rules for the first and the second class are presented below.

[*age* <=40] [*metastasis* = *no*] [*clinical aspect* = *unique nodule*] [*compressive syndromes* = *no*]; 0.364 ∪ [*metastasis* = *no*] [*pathological lymph nodes* = *no*] [*compressive syndromes* = *no*]; 0.468 ∪ [*histology* = *well differentiated*] [*metastasis* = *no*] [*clinical aspect* = *unique nodule*] [*invasion* = *no*]; 0.403
→ [class = class 1]
IP_GRE: $A_{learning}$ = 100 %, by assumption.

[*metastasis* = *yes*] [*compressive syndromes* = *yes*]; 0.104 ∪ [*histology* = *poorly differentiated*] [*metastasis* = *yes*]; 0.156 ∪ [*clinical aspect* = *important enlargement*]; 0.078 ∪ [*histology* = *poorly differentiated*] [*compressive syndromes* = *yes*]; 0.104 ∪ [40 < *age* <=60] [*histology* = *poorly differentiated*] [*pathological lymph nodes* = *yes*]; 0.065 ∪ [*metastasis* = *yes*]; 0.234 ∪ [*pathological lymph nodes* = *yes*] [*compressive syndromes* = *yes*]; 0.039 ∪ [*clinical aspect* = *multi nodules*] [*clinical lymph nodes* = *yes*]; 0.052. → [class = class *2*]
IP_GRE: $A_{learning}$ = 100 %, by assumption.

Next, we assume, that the classification rules for elements belonging to class *l* must correctly describe *most of the examples* belonging to class *l*, $l=1,2$, at least $A_{learning}$ = 97.5%, by assumption (i.e. may not describe not more than two training examples). We use IP_GRE (with elements of a greedy algorithm) and IP_GA (with elements of a genetic algorithm). We used the mutation rate = 0.01, the crossover rate = 0.6, and the size of the population = 50.

The classification rules for the first class are presented below.

[*age* <=40] [*metastasis* = *no*] [*compressive syndromes* = *no*]; 0.364 ∪.. [*metastasis* = *no*] [*pathological lymph nodes* = *no*] [*compressive syndromes* = *no*]; 0.468 ∪ [*histology* = *well differentiated*] [*metastasis* = *no*] [*invasion* = *no*]; 0.455
→ [class = class 1]
IP_GRE: $A_{learning}$ = 97.5 % by assumption.

[*metastasis* = *no*] [*clinical lymph nodes* = *no*] [*invasion* = *no*]; 0.468 ∪ [*compressive syndromes* = *no*] [*invasion* = *average*]; 0.052 ∪ [*clinical aspect* = *unique nodule*] [*clinical lymph nodes* = *yes*]; 0.104 → [class = class 1]
IP_GA: $A_{learning}$ = 97.5 % by assumption.

Table 1. Some parameters describing the process of finding a classification rule for the first class

Algorithm	$A_{learning}$ %, by assumption	Number of iterations	Number of selectors in rule
IP_GRE	100 %	3	11
	at least 97.5 %	3	9
IP2_GA	at least 97.5 %	3	7

The classification rules for the second class are presented below.

[*metastasis* = *yes*] [*compressive syndromes* = *yes*]; 0.104 ∪ [*histology* = *poorly differentiated*] [*metastasis* = *yes*]; 0.156 ∪ [*clinical aspect* = *important enlargement*]; 0.078 ∪ [*histology* = *poorly differentiated*] [*compressive syndromes* = *yes*]; 0.104 ∪ [40 < *age* <=60] [*pathological lymph nodes* = *yes*]; 0.078 ∪ [*metastasis* = *yes*]; 0.234 ∪ [*pathological lymph nodes* = *yes*] [*compressive syndromes* = *yes*]; 0.039 ∪ [*clinical aspect* = *multi nodules*] [*clinical lymph nodes* = *yes*]; 0.052 → [class = class *2*]
IP_GRE: $A_{learning}$ = 97.5 % by assumption.

[40 < *age* <=60] [*pathological lymph nodes* = *yes*]0.078 ∪ [*metastasis* = *yes*]; 0.234 ∪ [*clinical aspect* = *important enlargement*]; 0.078 ∪ [*sex* = *male*] [*invasion* = *average*]; 0.039 ∪ [*invasion* = large]; 0.065 ∪ [*age* <=40] [*clinical aspect* = *multi nodules*]; 0.026 → [class = class *2*]
IP_GA: $A_{learning}$ = 97.5 % by assumption.

Table 2. Some parameters describing the process of finding a classification rule for the second class

Algorithm	$A_{learning}$ %, by assumption	Number of iterations	Number of selectors in rule
IP_GRE	100 %	8	15
	at least 97.5 %	8	14
IP2_GA	at least 97.5 %	6	9

Some parameters describing the process of classification the patients into the first or the second class are presented in Table 3. The ratio of correct classification decisions to the total number of decisions made was taken as the measure of classification accuracy, in percentage.

Table 3. Some parameters describing the process of classification the patients into the first or second class

Algorithm	$A_{learning}$ %, by assumption	Classification accuracy, achieved
IP_GRE	100 %	100 %
	at least 97.5 %	98.7 %
IP_GA	at least 97.5 %	98.7 %

As can be seen, the shortest classification rules were obtained by using the method IP_GA.

7.2. Solving a Coronary Heart Disease Problem

There are several factors of blood, both morphological as well as of plasma, that can indicate some illness. Only very basic blood examination is unfortunately so far widely considered. Meanwhile, for example, the blood viscosity changes due to some physical and psychical conditions of people, both ill and well. It has been found [cf. Dintenfas (1969, 1981)] that the blood viscosity and plasma viscosity are functions of cells aggregation and fibrin level.

For example, the blood viscosity of smokers or those being under stress also increases. Additionally, within the last few years it has been observed that the rate of sial acid is responsible for existence of negative electrical charges in blood and plasma. Many other relations among various factors are still under investigations.

It is difficult for medical doctors to determine the descriptions of diseases in the form of general decision rules, hence machine learning methods from examples can be applied. The task is to develop a knowledge base that contains sufficient information to diagnose the coronary heart disease. That is, to provide decision rules which are sufficient to describe the disease in terms of symptoms.

In our considered medical classification problem there are 90 examples, either ill or healthy persons, 60 of them have been chosen as a so-called training set and another 30 persons for testing in order to check how the methods work and to compare the results. Table 4 shows the data for two selected persons, one ill and the second healthy.

Table 4. Data of two selected people

No	Attribute	Patient no 9 (healthy)	Patient no 10 (ill)
1	lk1	4.42	4.99
2	lk2	7.90	7.20
3	lk3	16.00	15.00
4	lp1	1.64	1.35
5	lp2	2.80	1.50
6	agr	2.22	2.25
7	fil	46.40	17.40
8	fib	297.00	330.40
9	ht	45.00	45.00
10	sas	0.059	0.077
11	sak	0.056	0.057
12	ph	7.348	7.367

The following 12 blood factors (attributes) have been measured:

lk1 - blood viscosity for coagulation quickness 230/s,
lk2 - blood viscosity for coagulation quickness 23/s,
lk3 - blood viscosity for coagulation quickness 15/s,

lp1	- plasma viscosity for coagulation quickness 230/s,
lp2	- plasma viscosity for coagulation quickness 23/s,
agr	- aggregation level of red blood cells,
fil	- blood cells capacity to change shape,
fib	- fibrin level in plasma,
ht	- hematocrite value,
sas	- sial acid rate in blood serum,
sak	- sial acid rate in blood cells,
ph	- acidity of blood.

Due to requirements of the considered inductive methods the real values of the a_j attributes, $j = 1,...,12$, have been arranged into some number of groups. The application of statistical analysis on the set of healthy persons has allowed to obtain for each j-th attribute its average value $\bar{p}_j$, a standard deviation δ_j and an interval $[\bar{p}_j - \delta_j, \bar{p}_j + \delta_j]$ that constituted three aggregated groups. Physicians have suggested four groups for two attributes: *sas* and *sak*, because of a higher accuracy required. This aggregated groups used for computation are shown in the Table 5. In this way the continuous-value problem has been changed into discrete-value one.

Table 5. The aggregated groups used for computation

No	Attribute	Group no 1	Group no 2	Group no 3	Group no 4
1	lk1	< 3.8	[3.8, 4.3]	> 4.3	
2	lk2	< 5.7	[5.7, 6.7]	> 6.7	
3	lk3	< 13	[13, 16]	> 16	
4	lp1	< 1.3	[1.3, 1.5]	> 1.5	
5	lp2	< 1.5	[1.5, 1.9]	> 1.9	
6	agr	< 1.7	[1.7, 2]	> 2	
7	fil	< 14.5	[14.5, 2.3]	> 22.3	
8	fib	< 212	[212, 298]	> 298	
9	ht	<42	[42, 46]	> 46	
10	sas	< 0.06	[0.06, 0.07]	(0.07, 0.08]	> 0.08
11	sak	< 0.06	[0.06, 0.07]	(0.07, 0.08]	> 0.08
12	ph	< 7.3	[7.3, 7.35]	> 7.35	

The problem of learning from examples is formulated as to find classification rules into the classes:

class 1: the patients have no coronary heart disease, are healthy,
class 2: the patients have a coronary heart disease, are ill,

using all examples as patterns.

The sets of examples corresponding to the groups were the input to the learning programs. We use IP_GA (with elements of a genetic algorithm) and IP_GRE (with elements of a greedy algorithm), and assume that the classification rules must correctly describe most of the learning examples belonging to *class 1* and *class 2*, at least $A_{learning}$, by assumption.

The ratio of correct classification decisions to the total number of decisions made was taken as the measure of classification accuracy, in percentage. The results of applying the methods to medical data are presented and described in Tables 6, 7 and 8.

Table 6. Some parameters describing the process of finding a classification rule for the first class

Algorithm	$A_{learning}$ %, by assumption	Number of iterations	Number of selectors in rule
IP_GRE	100 %	16	43
	at least 97 %	13	26
IP_GA	100 %	18	46
	at least 90 %	13	26

Table 7. Some parameters describing the process of finding a classification rule for the second class

Algorithm	$A_{learning}$ %, by assumption	Number of iterations	Number of selectors in rule
IP_GRE	100 %	19	55
	at least 97 %	17	33
IP_GA	100 %	19	49
	at least 90 %	19	37

The percentage of correct classifications is the measure of classification accuracy, in percentage. Better classification accuracy for testing examples was obtained by using the classification rules correctly describes most of the training examples.

Table 8. Some parameters describing the process of classification the patients into the first or second class

Algorithm	$A_{learning}$ %, by assumption	Classification accuracy, achieved
IP_GRE	100 %	90.0 %
	at least 97 %	96.7 %
IP_GA	100 %	73.4 %
	at least 90 %	96.7 %

We found the classification rule $R_W^{1*} = C_W^{l_1}; q(C_W^{l_1}) \cup \ldots \cup C_W^{l_L}; q(C_W^{l_L})$, where $l_1, \ldots, l_L \subseteq \{1, \ldots, K\}$, $q(C_W^{l_l})$, $l = 1, \ldots, L$ is the weight of the complex $C_W^{l_l}$ obtained by using (31). For each attribute a_j, $j=1, \ldots, K$ and each value $v_{j,i(j,l)} \in V_{a_j}$, we determine

$$U([a_j = v_{j,i(j,l)}]) = \sum_{l=1}^{L} SD([a_j = v_{j,i(j,l)}], C_W^{l_l})$$

where:

$$SD([a_j = v_{j,i(j,l)}], C_W^{l_l}) = \begin{cases} q(C_W^{l_l}) & \textit{if } [a_j = v_{j,i(j,l)}] \textit{ occurs in the complex } C_W^{l_l} \\ 0 & \textit{otherwise} \end{cases}$$

For each attribute a_j, $j=1, \ldots, K$ we determine $u(a_j)$

$$u(a_j) = \sum_{v_{j,t(j,l)} \in V_{a_j}} U([a_j = v_{j,t(j,l)}]) = \sum_{v_{j,t(j,l)} \in V_{a_j}} \sum_{l=1}^{L} SD([a_j = v_{j,t(j,l)}], C_W^{I_l})$$

Values $u(a_j), j=1, ..., K$ for classification rule for the first class (well) and for the second class (ill) are presented in the Table 9 and Fig. 18.

Table 9. Values $u(a_j), j=1, ..., K$ for classification rule for the first class (healthy) and for the second class (ill), IP_GRE method was applied, $A_{learning}$ = 97%

Attribute a_j	$u(a_j)$ for classification rule for the first class	$u(a_j)$ for classification rule for the second class	Total
lk1	0.017	0.050	0.067
lk2	0.0	0.0	0.0
lk3	0.266	0.250	**0.516**
lp1	**0.300**	0.200	**0.500**
lp2	0.100	**0.516**	**0.616**
agr	0.017	0.050	0.067
fil	0.0	0.033	0.033
fib	**0.300**	**0.266**	**0.566**
ht	0.083	0.017	0.100
sas	**0.450**	**0.350**	**0.800**
sak	0.133	0.266	0.399
ph	0.183	0.150	0.333

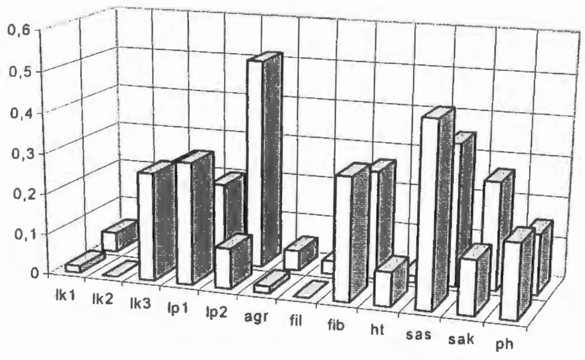


Figure 19. Values $u(a_j), j=1, ..., K$ (i.e. u(*lk1*),..., u(*ph*)) for classification rule for the first class (healthy) and for the second class (ill); IP_GRE method was applied.

8. Concluding Remarks

We have presented an improved inductive learning method IP with elements of genetic or greedy algorithms to derive classification rules from sets of positive and negative examples. The computational results are very encouraging.

References

Baker J.E. (1987) Reducing bias and inefficiency in the selection algorithm. In Genetic Algorithms and Their Applications: Proceedings of the 2nd International Conference on Genetic Algorithms. ed. J. J. Grefenstette, pp. 14-21, LEA, Cambridge, MA.

Balas E. (1980) Cutting planes from conditional bounds: a new approach to set covering. *Mathematical Programming Study* 12, 19-36.

Balas E. and Ho A. (1980) Set covering algorithms using cutting planes, heuristics and sub gradient optimisation: A computation study. *Mathematical Programming Study* 12, 37-60.

Balas E. and Padberg M.W. (1979) Set partitioning - A survey. In: N. Christofides (ed.) *Combinatorial Optimisation*, Wiley, New York.

Beasley J.E. (1987) An algorithm for set covering problem. *European Journal of Operational Research* 31, 85-93.

Beasley J.E. (1990) A Lagrangian heuristic for set covering problem. *Naval Research Logistics* 37, 151-164.

Beasley J.E. and Jornsten K. (1992) Enhancing an algorithm for set covering problem. *European Journal of Operational Research* 58, 293-300.

Beasley J.E., Chu P.C. (1994) *A genetic algorithm for the set covering problem.* Technical Report, The Management School, Imperial College.

Beasley J.E. (1996) A genetic algorithm for the set covering problem. *European Journal of Operational Research* 94, 392-404.

Christofides N. and Korman S. (1975) A computational survey of methods for the set covering problem. *Management Science* 21, 591-599.

Chvatal V. (1979) A greedy heuristic for the set-covering problem. *Math. of Operational Research* 4 (3) 233-235.

Croall I.F. and Mason J.P. (eds.) (1991), *Industrial Applications of Neural Networks.* Springer-Verlag, Berlin, 1991.

Etcheberry J. (1977) The set covering problem: A new implicit enumeration algorithm. *Operations Research* 25, 760-772.

Garfinkel R. S. and Nemhauser G.L. (1978) *Integer programming.* John Wiley & Sons, New York-London-Sydney-Toronto.

Grefenstette John J.(1990) *User's guide to GENESIS.*

Grossman T. and Wool A. (1995) Computational experience with approximation algorithms for the set covering problem. Working paper, Theoretical Division and CNLS, Los Alamos National Laboratory.

Homaifar A., Lai S., Qi X. (1994) Constrained optimization via genetic algorithms. Simulation, vol. 62, s. 242-254.

Jacobs L.W. and Brusco M.J. (1993) A simulated annealing-based heuristic for the set-covering problem. Working paper, Operations Management and Information Systems Department, Northern Illinois University, Dekalb, IL.

Jefries C. (1991) *Code Recognition and Set Selection with Neural Networks.* Birkhauser, Boston.

Johnson D.A. (1974) Approximation algorithms for combinatorial problems. *J. Computer System Sci.* 9, 256-278.

Joines J.A., Houck C.R. (1994) On the use of non-stationary penalty functions to solve nonlinear constrained optimization problem with Gas. In: Michalewicz Z., Schaffer D., Schwefel H.P., Fogel D., Kitano H. (red.): Proceedings of the First IEEE

International Conference on Evolutionary Computation, IEEE Service Center, Piscataway, NJ, vol.2, Orlando, 27-29 June, s. 579 - 584.

Kacprzyk J. and Iwanski C. (1992) Fuzzy logic with linguistic quantifiers in inductive learning, In: L.A. Zadeh and J. Kacprzyk (Eds.), *Fuzzy Logic for the Management of Uncertainty*, Wiley, pp. 465- 478.

Kacprzyk J., Szkatula G. (1994a) Machine learning from examples under errors in data, Proceedings of Fifth International Conference in Information Processing and Management of Uncertainty in Knowledge-Based Systems IPMU'94 Paris France, Vol. 2, pp. 1047 – 1051.

Kacprzyk J. and Szkatula G. (1994b) An approach to inductive learning under errors in data, Proceedings of ANZIIS'94 Conference, Brisbane, Australia, pp. 352 – 356.

Kacprzyk J., Szkatula G. (1995): Machine learning from examples under errors in data. In B. Bouchon-Meunier, R.R. Yager and L.A. Zadeh (eds.): *Fuzzy Logic and Soft Computing*, World Scientific, Singapore, pp. 31 – 36.

Kacprzyk J. and Szkatuła G. (1996): An algorithm for learning from erroneous and incorrigible examples, *International Journal of Intelligent Systems*, vol 11, pp. 565 – 582.

Kacprzyk J. and Szkatula G (1997a) An improved inductive learning algorithm with a preanalysis of data, In: Z.W. Raś and A. Skowron (eds.): *Foundations of Intelligent Systems* (Proceedings of 10th ISMIS'97 Symposium, Charlotte, NC, USA), Springer-Verlag, Berlin, pp. 157 - 166.

Kacprzyk J. and Szkatula G. (1997b) Deriving IF-THEN rules for intelligent decision support via inductive learning", in N.Kasabov et al. (eds.): Progress in Connectionist-Based Information Systems (Proceedings of ICONIP'97, ANZIIS'97 and ANNES'97 Conference, Dunedin, New Zealand), Springer, Singapore, vol. 2, pp. 818 - 821.

Kacprzyk J. and Szkatula G. (1998) IP1 - An Improved Inductive Learning Procedure with a Preprocessing of Data. Proceedings of IDEAL'98 (Hong Kong), Springer-Verlag, pp. 385-392.

Kacprzyk J. and Szkatula G. (1999) An inductive learning algorithm with a preanalysis of data. International Journal of Knowledge - Based Intelligent Engineering Systems, vol. 3, pp. 135-146.,

Kacprzyk J., Szkatula G. (2002a) An integer programming approach to inductive learning using genetic and greedy algorithms, In: L.C. Jain and J. Kacprzyk (eds.): *New Learning Paradigms in Soft Computing*, Physica-Verlag, Heidelberg and New York, pp. 322-366.

Kacprzyk J., Szkatula G. (2002b) An integer programming approach to inductive learning using genetic algorithm. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02, Honolulu, Hawaii, pp. 181-186.

Kacprzyk J., Szkatuła G. (2005a) A softened formulation in inductive learning and application to coronary disease data. In: Atanassov K., Kacprzyk J., Krawczak M., Szmidt E. (Eds.): *Issues in the Representation and Processing of Uncertain and Imprecise Information. Fuzzy Sets, Intuitionistic Fuzzy Sets, Generalized Nets, and Related Topics.* AOW EXIT, Warszawa, pp. 181-197

Kacprzyk J., Szkatuła G. (2005b) A softened formulation of inductive learning and its use for coronary disease data. *Lecture Notes in Artificial Intelligence*, vol. 3488, pp. 200-209.

Kacprzyk J., Szkatuła G. (2005c) An inductive learning algorithm with a partial completeness and consistence via a modified set covering problem. *Lecture Notes in Computer Science*, vol. 3697, pp. 661-666.

Lovasz L. (1975) On ratio of optimal integral and fractional covers. *Disc. Math.* 13, pp. 383-390.

Michalski R. S. (1983) A theory and methodology of inductive learning. In: R. Michalski, J. Carbonell and T.M. Mitchell (Eds.), Machine Learning. Tioga Press.

Michalski R.S., I. Mozetic, J. Hong and N. Lovrac (1986) The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. Proceedings of 5th Nat. Conference on Artificial Intelligence (Philadelphia). Morgan Kaufmann, pp. 1041-1045.

Nakache J.P., Asselain B. (1983) Medical data set proposed for the workshop on data analysis. EIASM Workshop, April 1983.

Paixao J. (1984) Algorithms for large scale set covering problems. PhD. Thesis, Department of Management Science, Imperial College, London SW7 2BX, UK

Powell D., Skolnick M.M. (1993) Using genetic algorithms in engineering design optimization with non-linear constraints, W: Forrest S. (red.): Proceedings of the Fifth International Conference on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp. 424 - 430.

Schoenauer M., Xanthakis S. (1993) Constrained GA optimization, W: Forrest S. (red.): Proceedings of the Fifth International Conference on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA, pp. 573 - 580.

Sen S. (1993) Minimal cost set covering using probabilistic methods. Proc. 1993 ACM/SIGAPP Symposium on Applied Computing, pp. 157-164.

Szkatuła G. (1995) Machine learning from examples under errors in data (In Polish), Ph.D. thesis, SRI PAS Warsaw, Poland.

Szkatuła G. (2002) Zastosowanie zmodyfikowanego zadania pokrycia w uczeniu maszynowym. In: Gutenbaum J. (ed.): Automatyka Sterowanie Zarządzanie. SRI PAS, Warsaw, pp. 431-445.

Szkatuła G., Kacprzyk J. (2005) An inductive learning algorithm with a partial completeness and consistency. In: Draminski M., Grzegorzewski P., Trojanowski T., Zadrozny S. (Eds.): Issues in intelligent systems. Models and techniques. Akademicka Oficyna Wydawnicza EXIT, Warszawa, pp. 229-246.

Thurn S. et al. (1991) The MONK's Problems. A Performance Comparison of Different Learning Algorithms. Carnegie-Mellon University, Rep. CMU-CS-91-197.

Vercellis C. (1984) A probabilistic analysis of the set covering problem. *Annals of Oper. Research* 1, pp. 255-271.

Wnek J., Sarma J., Wahab A., Michalski R, S. (1991) Comparison learning paradigms via diagrammatic visualization: A case study In single koncept learning using symbolic, neural net and genetic algorithm methods. Technical Reports, George Mason University, Computer Science Department.