v. 4, n. 23, 2024

Journal of
Engineering
Research

UIB-AILS: A MODEL FOR AUTONOMOUS INDUCTIVE LEARNING SYSTEMS

Gabriel Fiol-Roig
University of the Balearic Islands
Islas Baleares, Spain

Acceptance date: 20/09/2024

Abstract. Systems that learn autonomously constitute a relevant exponent with- in the category of intelligent systems. Such systems are characterized by the ability to update their decision/action hypothesis over time without any external intervention. Such capacity for updating is based on some main considerations, such as: the experiences accumulated by the system, the acquisition of new knowledge from the environment, the general structural characteristics of the hypothesis and the criterion specifying the particular properties that the hypoth- esis must satisfy. The design of an autonomous inductive learning system – AILS - is a complex task. This work presents a general model for the design of AILSs, whose components can be customized according to the nature of the problem in hand, so that the model is suitable for addressing the design of a variety of AILS with different peculiarities. Such a model adopts a cyclic evolutionary configuration that includes, among others, components to deal with data imprecision, tech- niques to handle the vagueness of decisions/actions, and methods to process in a unified way knowledge coming from different levels of abstraction, such as raw data and logic expressions.
Keywords: Evolutionary Inductive Learning Systems, Knowledge Acquisition, Qualitative Knowledge.

INTRODUCTION

Evolutionary learning is the process by which a system/agent improves its function over time [1]. In this way, a system/agent is said to learn if its behavior over time is consistent with the proposed objectives. Two important issues for an agent to be able to learn over time refer to: the agent's ability to reason about the knowledge provided by the environment and the experiences accumulated by the agent over time.

The design of autonomous inductive learning systems from classified examples is a complex task, since the issues affecting the learning process are numerous and some of them difficult to solve. In the literature there are hardly any models that cover a design of this type. Many of the published models simply cover the design of induc- tive reasoning systems, but they are not focused on the task of true inductive learning. In fact, the concepts of reasoning and learning are often confused.

This paper presents a model for autonomous evolutionary learning from examples and accumulated experiences by the system/agent up to that point. The learning pro- cess has no time limits, that is, it is a permanently open process of improving the per- formance of the system/agent. Thus, the decision/action capacity of the system/agent at a given time depends solely on the knowledge provided by the environment - ex- amples - and the experiences accumulated by the agent. Regarding knowledge pro- cessing, it should be emphasized that the proposed model is based on inductive- deductive reasoning [2], since it works simultaneously on specific knowledge struc- tures -the examples from the environment- and on abstract structures that give support to the decision/action hypothesis and the experiences -decision trees, logical expres- sions, graphs, neural networks, etc.- in order to improve the decision/action hypothe- sis. The components of the inductive model presented in this work are briefly described below.

The external *environment* is the device that provides external knowledge to the agent. In the particular case that concerns us, this knowledge is expressed in terms of concrete examples about some concept. It is specific knowledge that can include pre- cise and imprecise data. In [2, 3, 4] it has been proved that imprecise data can provide useful

qualitative knowledge for the decision/action hypothesis of the system/agent.

The *decision/action hypothesis*, also known as the *knowledge base*, contains the agent's knowledge about the environment – that is, the agent's interpretation of the environment. It has an abstract – intentional – format, which facilitates the sys- tem/agent's deductive reasoning in achieving its objectives. The decision/action hy- pothesis is updated over time.

The central core of the model, called *knowledge acquisition device*, is responsible for simultaneously processing specific knowledge from the environment and abstract knowledge from the hypothesis and experiences of the system/agent. The knowledge acquired is abstract and constitutes the current hypothesis or knowledge base. This device constitutes the inductive-deductive stage of the model.

The *fact base* stores the perceptions that the agent receives from the environment at each instant in time. Perceptions are used as facts by the process of deductive reason- ing -that is, the action-decision device- from the knowledge base. Perceptions are described by specific knowledge.

Based on the perceptions of the fact base, the *decision-action device* is responsible for reasoning deductively from the knowledge base. The results of this reasoning process are the decisions/actions that the agent will adopt.

The *execution-verification device* is responsible, on the one hand, for executing the decisions/actions deduced by the decision-action device and, on the other hand, for verifying their quality. The results of this checking process will constitute the sys- tem/agent's experiences.

The *feedback stage* determines the cyclical nature of the model. It is is based on the knowledge provided by the experiences about the decisions/actions taken. This knowledge is the result of the execution-verification device.

The main components of the model are described in more detail in the following sections.

AN AUTONOMOUS INDUCTIVE LEARNING SYSTEM MODEL

This section presents a model for autonomous inductive learning from classified ex- amples. This model is an extension of the simple model described in [1]. Figure 1 illustrates such model.

From Figure 1 it can be seen that this is a cyclic model, where the knowledge base and the system experiences provide feedback to the model, in particular to the knowledge acquisition device.

The model consists of the six devices that have been briefly described in the previ- ous section, and the feedback stage. In order to clarify the exposition of this work, two types of feedback are distinguished in Figure 1:

- *Feedback 1* is the simplest, as it only considers the knowledge base as the knowledge element from which the knowledge acquisition device receives feed- back - note in Figure 1 that the experiences of the system/agent have not been con- sidered as feedback elements-. It is important to emphasize again that the knowledge base is expressed in an abstract format.

- *Feedback 2* is the complete feedback, as discussed in the previous section, cover- ing the knowledge base and the system/ agent's experiences, both expressed in an abstract format.

This is an evolutionary and incremental model that works by cycles, so that each cy- cle defines a new stage of the system/agent. Each cycle must necessarily lead to an improvement of the knowledge base. Each cycle is referred to as a *learning cycle*.

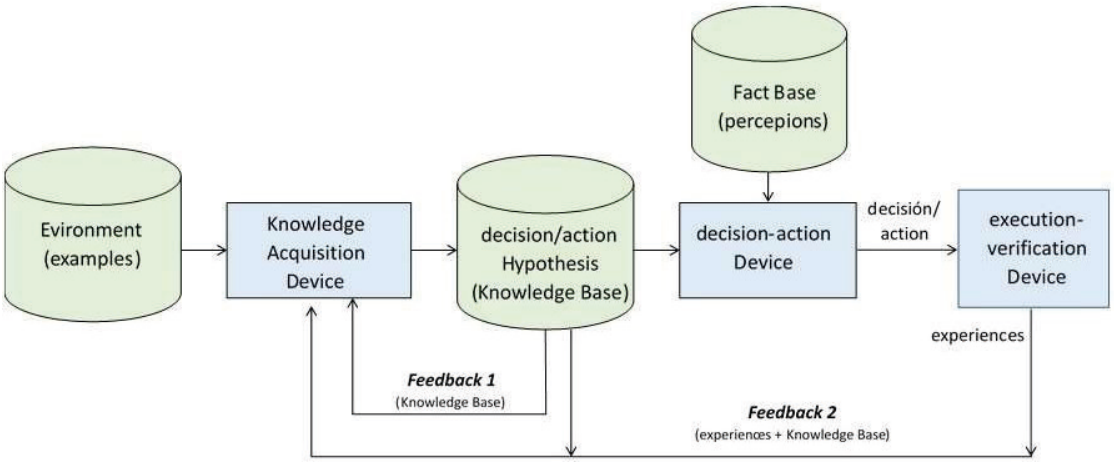

Fig. 1. Model for autonomous inductive learning from classified examples.

The number of cycles of the system/agent is limited by the number of examples and experiences over time that the environment and the execution-verification device are able to provide to the model. That is, as long as the environment and/or the execution-verification device provide examples/ experiences to the model, the system/agent will process them in order to improve the knowledge base. The knowledge base is updated at each learning cycle, according to the examples and the experiences gener- ated up to the given cycle. This model is very useful in complex environments, where the scope of the number of examples is not known. In this way, the learning process of the system/agent can work indefinitely.

THE ENVIRONMENT

Also referred to as Object-Attribute Table -abbreviated OAT-, it constitutes the exter- nal input knowledge of the knowledge acquisition device [2, 5, 6]. Such knowledge is expressed in the form of examples or concrete portions of knowledge about some concept. The notion of OAT is referred to as Information System by other authors [7], however there are relevant differences between both concepts. Next the concept of OAT is formally described.

An OAT consists of a set of classified examples and a set of attributes in terms of which the examples are described. An OAT is said to be completely specified if the values of the examples with respect to each of the attributes are known and fully spec- ified [2, 8]. On the other hand, an OAT is incompletely specified if some of the attrib- ute values are not completely specified, or in the extreme case, they are completely unknown [3, 4].

Completely and incompletely specified OATs are processed in a unified way by the knowledge acquisition device, that is, there is no need to distinguish between both types of OAT. The concept of OAT is formally defined below.

Definition 1 (representing attribute values). The value of an attribute r_j in a specific situation di can be represented by a subset $V_{di,rj}$, $V_{di,rj} = \{v^i_{j1}, v^i_{j2}, \ldots, v^i_{jp}\}$, being $v^i_{j1}, v^i_{j2}, \ldots, v^i_{jp}$ the different simple values adopted by r_j in the situation d_i. If the number of values of $V_{di,rj}$ is greater than 1, that is $\#V_{di,rj} > 1$ -where the symbol # represents the cardinal of $V_{di,rj}$ -, then ri is said to be an incompletely specified/vague attribute. If $\#V_{di,rj} = 1$ then r_i is a completely specified attribute. $\#V_{di,rj}$ can never be less than 1. If $V_{di,rj}$, $V_{di,rj} = \{v^i_{j1}, v^i_{j2}, \ldots, v^i_{jp}\}$ is a vague value of attribute r_j -that is, $\#(V_{di,rj}) > 1$- in the specific situation d_i, then $V_{di,rj}$ will also be represented by the symbol $*^i_{j1,j2,\ldots,jp}$.

According to definition 1, let us note that if an attribute r_j is vague in any specific situation d_i, then r_j is definitely vague - that is, r_j is considered simply a vague attribute -. If there is no specific situation d_i in which r_j is vague, then r_j is definitely a fully specified attribute.

Definition 2 (domain of an attribute). The domain V_i of the attribute ri covers the values that the attribute can take in all specific situations observed up to the current time. This means that:

- Since V_i is made up of values of r_i, then V_i is a set of subsets, since, according to definition 1, each value is represented by a set.
- As V_i is made up of the values of r_i observed up to "the current time", then V_i may evolve over time, since new values of r_i will possibly be observed in the future.

Definition 3 (Object Attribute Table). OAT= < D, R, W, V, V', F, H, Q, C, f >, where:

$D = \{d_1, d_2, \ldots, d_m\}$ is a set of examples or concrete portions of knowledge.

$R = \{r_1, r_2, \ldots, r_n\}$ is a set of qualities or attributes in terms of which examples of D are described.

$W=\{W_1, W_2, \ldots, W_n\}$ is A set of domains of simple values of the attributes, r_i, $r_i \in R$, i=1…n, being, W_j, $W_j \in W$, the domain of simple values of attribute r_j in the OAT. A simple domain is made up of all possible simple values of attributes.

$V=\{V_1, V_2, \ldots, V_n\}$ is a set consisting of the domains of each attribute, r_i, $r_i \in R$, i=1…n, being, $V_j=\{ v_1,r_j, v_2,r_j,\ldots, v_{p,rj}\}$, $V_j \in V$, the domain of attribute r_j in the OAT. Note that the values of a domain Vi are all different and will take the form $v_{k,ri}$, $v_{k,ri} \in V_i$. According to definition 1, each value $v_{k,ri} \in V_i$ is a subset of simple values of W_i.

C is a set of concepts, $C = \{C_1, C_2,\ldots, C_w\}$. Each C_i i=1…w, represents a concept f is a function that assigns to each element of D its corresponding concepts, that is, f: D→Π(C), where Π(C) denotes the set of parts of C. Note that each example, $d_i \in D$, has a subset of concepts associated.

$V', V' = \{V'_1, V'_2, \ldots, V'_n, V'_{1,2}, \ldots, V'_{i,j,\ldots k},\ldots, V'_{1,2,\ldots n}\}$, represents the set of do- mains of all attribute-value tuples of the OAT corresponding to all attribute subsets of R. That is, $V'_{i,j,\ldots,k}$, with i=1…n, j=1…n,… k=1…n, i≠j≠…≠k, represents the domain of attribute-value tuples of the OAT corresponding to the attribute subset $\{r_i, r_j,\ldots, r_k\}$. Note how set V' constitutes an extension of V to all attribute subsets of the OAT. Set V' contains 2^n-1 elements - that is, domains of attribute-value tuples.

$F=\{f_1, f_2, \ldots, f_n\}$, $f_i : Dx\{r_i\}\rightarrow V_i$, i=1...n, is a set of functions that define the values adopted by the elements of D for each attribute r_i, i = 1, 2, …, n.

$H=\{h_1, h_2,...h_n\}$ is a set of functions, one for each attribute ri of the OAT, such that h_j, j=1..n, associates the corresponding subset of concepts to each simple value v_{jk}, $v_{jk} \in v_{w,rj}$, $v_{w,rj}=\{v_{j1}, v_{j2},\ldots, v_{jp}\}$, $vw_{,rj} \in V_j$. That is, $h_j:W_j\rightarrow\Pi(C)$, so that $h_j(v_{jk}) = f(d_1) \cup f(d_2) \cup\ldots\cup f(d_m)\}$ for all d_s, s=1..m, such that $v_{jk} \in f_j(d_s, r_j)$.

$Q=\{q_1, q_2,...q_n\}$ is a set of functions, one for each attribute r_i of the OAT. Let $v_{k,rj}=\{v_{j1}, v_{j2},\ldots, v_{jp}\}$, $v_{k,rj} \in V_j$, be a value of attribute r_j, where v_{jk}, k=1..p, are its sim- ple values. Functions of set Q take the form q_j: $V_j\rightarrow\Pi(\Pi(C))$, so that each q_j, j=1..n, associates the corresponding set of subsets of concepts of C to each different value $v_{k,rj}=\{v_{j1}, v_{j2},\ldots, v_{jp}\}$, $v_{k,rj} \in V_j$, being $q_j(v_{k,rj})=\{h_j(v_{j1}), h_j(v_{j2}),\ldots, h_j(v_{jp})\}$. Note that expression $\{h_j(v_{j1}), h_j(v_{j2}),\ldots, h_j(v_{jp})\}$ represents a set of subsets of concepts, one for each simple value, v_{jk}, $v_{jk} \in v_{k,rj}$. It is important to emphasize that the sets defined by the functions $q_j(v_{k,rj})=\{h_j(v_{j1}), h_j(v_{j2}),\ldots, h_j(v_{jp})\}$ may contain repeated elements -that is, repeated subsets of concepts of C-. To avoid confusion in practice, it is important to define an order relationship between the subsets defined by the functions $q_j(v_{k,rj})$.

Figure 2 illustrates graphically an example of an OAT.

Example. Consider the OAT of Figure 2. According to definitions 1, 2 and 3, we have:

D = {d_1, d_2, d_3, d_4, d_5, d_6, d_7} is a set of examples.

R = {r_1, r_2, r_3, r_4} is the set of attributes in terms of which the examples are de- scribed.

C = {C_a, C_b, C_d} is the set of concepts.

The right column of the table, represented by the symbol Π(C), represents the con- cepts of C associated with each element di ϵ D.

D\R	r_1	r_2	r_3	r_4	Π(C)
d_1	{0}	{red}	{1}	{0}	{C_a}
d_2	{0}	{green}	{0}	{1}	{C_b}
d_3	{$*^1_{0,1}$}	{red}	{0}	{1}	{C_b}
d_4	{0}	{blue}	{0}	{$*^4_{0,1}$}	{C_d}
d_5	{1}	{blue}	{2}	{0}	{C_a}
d_6	{0}	{red}	{$*^3_{0,1,2}$}	{1}	{C_b}
d_7	{1}	{blue}	{1}	{1}	{C_a}

Fig. 2. Example of an OAT.

$V_{di,rk}$, $V_{di,rk}$ ϵ V_i, i=1…m. k=1…n, are the values of the examples for each attribute. These values are actually subsets of values -see definition 1-, so they are enclosed in parentheses. Note how element d_3 takes, with respect to attribute r_1, the value Vd3,r1={$*^1_{0,1}$}, which can also be represented as $V_{d3,r1}$= {0, 1}. Such a value is a vague value, since #$V_{d3,r1}$>1. The same occurs with values $V_{d6,r3}$= {0, 1, 2} and $V_{d4,r4}$={0, 1}. The rest of values in the OAT are fully specified.

W={W_1, W_2, …, W_n} is a set of domains of simple values of the attributes, r_i, r_i ϵ R, i=1…n. Thus, W_1={0, 1}, W_2={red, green, blue}, W_3={0, 1, 2}, W_4={0, 1}.

V = {V1, V2, V3, V4}, represents the set of attribute domains of R, where V_i is the domain of attribute r_i. Thus, we have: V_1={{0}, {1}, {$*^1_{0,1}$}}={{0}, {1}, {0,1}}, V_2={{red}, {green}, {blue}}, V_3={{0}, {1}, {2}, {$*^3_{0,1,2}$}}={{0}, {1}, {2}, {0,1,2}}, V_4 = {{0}, {1}, {$*^4_{0,1}$}}={{0}, {1}, {0,1}}.

V' = {V'_1, V'_2, V'_3, V'_4, $V'_{1,2}$, $V'_{1,3}$, $V'_{1,4}$, $V'_{2,3}$, $V'_{2,4}$, $V'_{3,4}$, $V'_{1,2,3}$, $V'_{1,2,4}$, $V'_{1,3,4}$, $V'_{2,3,4}$, $V'_{1,2,3,4}$}, represents the set of domains of attribute-value tuples of the OAT, for all attribute subsets of R. Some examples of such domains are:

V'_1={<0>, <1>, <$*^1_{0,1}$>},

$V'_{1,2}$={<0, red>, <0, green>, <$*^1_{0,1}$, red>, <0, blue>, <1, blue>},

$V'_{2,3,4}$={<red, 1, 0>, <green, 0, 1>, <red, 0, 1>}, <blue, 0, $*^4$0,1>, <blue, 2, 0>, <red, $*^3_{0,1,2}$, 1>, <blue, 1, 1> }.

F={f_1, f_2, f_3, f_4} is a set of functions that define the values adopted by the elements of D for each attribute r_i, Examples:

$f_1(d_1, r_1)$ = {0}, $f_1(d_2, r_1)$ = {0}, $f_1(d_3, r_1)$ = {$*^1_{0,1}$};

$f_2(d_1, r_2)$ = {red}; $f_2(d_2, r_2)$ = {green}; $f_2(d_3, r_2)$={red}.

H={h_1, h_2,...h_n} is a set of functions h_i, h_j:W_j→Π(C), one for each attribute r_i of the OAT, such that h_j, j=1..n, associates the corresponding set concepts of C to each simple value v_{jk}, v_{jk} ϵ $v_{w,rj}$, vw,rj={v_{j1}, v_{j2},…, v_{jp}}, $v_{w,rj}$ ϵ V_j, so that $h_j(v_{jk})$=f(d1)uf(d2) ∪…∪ f(d_m)} for all d_s, s=1..m, such that v_{jk} ϵ $f_j(d_s, r_j)$.

Examples:

Let us consider attribute r1. W_1={0, 1}. Then $h_1(0)$={C_a}∪{C_b}}∪{C_b}u {C_d}}∪{C_b}={C_a, C_b, C_d}; $h_1(1)$={C_b}∪{C_a}∪{C_a}={C_a, C_b}.

Let now consider attribute r3. W_3={0,1,2}. Then $h_3(0)$={C_b}∪{C_b}∪{C_d}∪{C_b}={C_b, C_d}; $h_3(1)$={C_a}∪{C_b}∪{C_a}={C_a,C_b}; $h_3(2)$={C_a}∪{C_b}={C_a, C_b}.

Q={q_1, q_2,...q_n} is a set of functions q_j:V_j→Π(Π(C)), one for each attribute r_i of the OAT, so that each q_j, j=1..n, associates the corresponding set of subsets of concepts of C to each different value $_{vk,rj}$={v_{j1}, v_{j2},…, v_{jp}}, $v_{k,rj}$ ϵ V_j, being $q_j(v_{k,rj})$={$h_j(v_{j1})$, $h_j(v_{j2})$,…, $h_j(v_{jp})$}.

Examples:

Consider domain V_1={{0},{1},{0,1}} of attribute r1. Then q_1{{0}}= {h1(0)}= {C_a, C_b, C_d}; q_1{{1}}= {h_1(1)}= {C_a, C_b}; q_1{{0,1}}= {h_1(0), h_1(1)}= {{C_a, C_b, C_d }, {C_a, C_b}}.

Let now consider domain V_3={{0}, {1}, {2}, {0,1,2}} of attribute r_3. Then q_3{{0}}= {h_3(0)}= {C_b,C_d}; q_3{{1}}= {h_3(1)}= {C_a,C_b}; q_3{{2}}= {h_3(2)}= {C_a,C_b}; q_3{{0,1,2}}= {h_3(0), h_3(1), h_3(2)}= {{C_b, C_d}, {C_a, C_b}, {C_a, C_b}}.

Due to reasons of space and clarity in the presentation of concepts, the OAT de- scribed in this section has been formulated only for attributes with discrete domains. An extension for the case of OAT with continuous attributes is presented in [4].

THE KNOWLEDGE ACQUISITION DEVICE

The knowledge acquisition device -abbreviated KAD- is responsible for updating the knowledge base -abbreviated KB- in each learning cycle of the system/agent, so that the resulting KB is the one that best fits the established criterion. This criterion de- pend on the nature of the problem being solved and specifies the characteristics that the KB must satisfy.

Let us first consider the Feedback-1 of the learning model in Figure 1. The key to updating the KB is to design a qualitative knowledge acquisition method from new examples coming from the Environment and from the KB, in order to improve the functionality of the KB itself. If we now consider Feedback-2 from Figure 1, then the qualitative knowledge acquisition method just mentioned must consider, in addition to new examples from the environment and the KB, the experiences of the system/agent. From the just mentioned considerations in this paragraph, we can say that the referred qualitative knowledge acquisition method constitutes the core of KAD.

Due to space restrictions and in order to clarify the discussion, only the learning model based on Feedback-1 will be presented in this section.

For practical reasons, the decision tree structure has been considered as the format to support the knowledge base -KB-. A decision tree can be interpreted as a set of antecedent-consequent rules, such that each branch of the tree from the root to a leaf node constitutes a rule. In turn, each example from the Environment can also be seen as an antecedent-consequent rule. This is the first step: converting all the input knowledge to the KAD -examples and KB- into the same abstract format -antecedent- consequent rules-, which requires an appropriate conversion process. The next step is to reason about the resulting set of antecedent-consequent rules, in order to modify the KB, which adopts a decision tree format. This is a process of extraction of qualita- tive knowledge from a set of rules in order to obtain a decision tree. Such a process must necessarily be guided by a criterion which defines the characteristics that the final decision tree must satisfy.

The most relevant issues of the above two-step process carried out by the KAD will be described below.

INDUCTIVE INFERENCE ON AN OAT: ATTRIBUTE BASES

The selection of an «adequate» subset of attributes, R_x, $R_x \subseteq R$, to intensively de- scribe the subsets or concepts of C in an OAT, constitutes one of the main stages of the inductive inference process. The term «adequate» means that the attributes of R_x must allow to describe the concepts of C according to the characteristics of the crite- rion considered. Such characteristics, independently of the nature of the particular problem, must guarantee, in any case, an accurate description of the concepts, that is, free of confusions or contradictions.

Any subset of attributes R_x, $R_x \subseteq R$, that guarantees a confusion-free description of the concepts of C is called an attribute basis of R

with respect to C. Much of the com- putational effort of an inductive process will focus on the search for some "adequate" attribute basis, particularly when it comes to getting an optimal basis. Complete doc- umentation on the concept of attribute basis can be found in [2, 3, 4, 5, 8].

The following provides a formal definition of the concept of attribute basis.

Definition 4 (couples of partially common values). Consider a given OAT. Let $v_{k,rj}, v_{s,rj} \in V_j$, $k \neq s$, be two values in the domain of the same attribute r_j. If it holds that $v_{k,rj} \cap v_{s,rj} \neq \emptyset$, then $v_{k,rj}$ and $v_{s,rj}$ are said to be a couple of partially common values. Note that partially common values are defined on the same domain.

Definition 5 (contradictory individual values). Consider a given OAT. Let $r_j \in R$ be an attribute of R with domain V_j. Let $v_{k,rj}=\{v_{j1}, v_{j2},\ldots, v_{jw}\}$ be a value in the domain V_j, $v_{k,rj} \in V_j$. $v_{k,rj}$ is said to be a contradictory value if and only if there exists some $d_r \in D$, $f_j(d_r,r_j) \cap v_{k,rj} \neq \emptyset$, such that $f(d_r) \subset h_j(v_{js})$, s=1..w, for any simple value $v_{js} \in v_{k,rj}$, $v_{js} \in f_j(d_r,r_j) \cap v_{k,rj}$ –Note that the symbol $\subset$ refers to the strict inclusion of subsets -.

Definition 6 (couples of contradictory values). Consider a given OAT. Let $r_j \in R$ be an attribute of R with domain V_j. Let $v_{k,rj}=\{v_{j1}, v_{j2},\ldots, v_{jw}\}$ and $v_{p,rj}=\{v'_{j1}, v'_{j2},\ldots, v'_{js}\}$ be to values in the domain V_j, $v_{k,rj} \in V_j$, $v_{p,rj} \in V_j$, such that $v_{k,rj} \cap v_{p,rj} \neq \emptyset$. Let $d_a \in D$ and $d_b \in D$, $a \neq b$, be such that $f_j(da, r_j)=v_{k,rj}$ and $f_j(d_{b,\ rj})=v_{p,rj}$. If $f(d_a) \neq f(d_b)$ then the cou- ple $(v_{k,rj}, v_{p,rj})$ is said to be a couple of contradictory values. In the case that $v_{k,rj}=v_{p,rj}$, then $v_{k,rj}$/ $v_{p,rj}$ are individual contradictory values -see definition 5-.

Definition 7 (couples of contradictory attribute-value tuples). Consider a given OAT. Let $e_a = <V_{dv,ri}, V_{dv,rj},\ldots, V_{dv,rk}>$, $e_a \in V'_{i,j,\ldots k}$, $V_{dv,rs} \in Vs$, s=i,j,...,k, and $e_b=<V_{dw,r}i, V_{dw,rj}, \ldots, V_{dw,rk}>$, $e_b \in V'_{i,j,\ldots k}$, $V_{dw,rp} \in V_p$, p=i,j,…k, v≠w, v,w = 1…m, be a couple of two attribute-value tuples defined on the same attribute subset Rx={ ri, rj,…, rk}, Rx⊂R. Tuples ea and eb are said to be a couple of contradictory tuples with respect to the subset of attributes R_x if and only if [(($V_{dv,ri}$, $V_{dw,ri}$), ($V_{dv,rj}$, $V_{dw,rj}$), …, ($V_{dv,rk}$, $V_{dw,rk}$) are all couples of partially common values) AND ($f(d_v) \neq f(d_w)$)].

Definition 8 (attribute basis). Consider a certain OAT. Let $R_x=\{r_i, r_j,\ldots, r_k\}$, $R_x \subseteq R$, be a subset of attributes of the OAT. R_x is said to be an attribute basis with respect to C if, and only if, there exist no couples of contradictory attribute-value tuples with respect to the subset R_x.

The concept of attribute basis is fundamental in induction from examples, since the knowledge base is expressed in terms of an attribute basis.

Example. Consider the OAT of Figure 2. Let us test wether subset {r2, r3} is an at- tribute basis. According to definition 8, {r2, r3} is an attribute basis with respect to C if, and only if, there is no couples of contradictory attribute-value tuples with respect to the subset {r2, r3}. Let's check if this statement is true.

Consider tuples $e_a = <V_{d1,r2}, V_{d1,r3}>$ and $e_b= <V_{d2,r2}, Vd_{2,r3}>$, whose values in the OAT are $e_a=<\{red\},\{1\}>$ and $e_b=<\{green\},\{0\}>$. Reasoning according to definition 7, we see that the couple ({red}, {green}) is not a couple of partially common value. Therefore, ea and eb are not a pair of contradictory attribute-value tuples, so we cannot claim that ea and eb are not an attribute base.

Consider now tuples ea = $<V_{d1,r2}, V_{d1,r3}>$ and $e_b = <V_{d6,r2}, V_{d6,r3}>$, with values $e_a=<\{red\},\{1\}>$ and $e_b=<\{red\},\{0,1,2\}>$ respectively. We have that the couples ({red}, {red}) and ({1}, {0,1,2}) are both couples of partially common values, since {red}∩{red}≠Ø and {1}∩{0,1,2}≠Ø. Now we have tu check the second condition of definition 7. We can see in Figure 1 that $f(d_1)=\{C_a\}$ and $f(d_6)=\{C_b\}$, thus $f(d_1) \neq f(d_6)$. We have just shown that {r2, r3} is not an attribute basis.

Let us test now whether subset {r2, r3, r4} is an attribute basis.

If we look at all the couples of attribute-value tuples in set R_x={r2, r3, r4} of the OAT, we see that the only couples of tuples that satisfies the condition «(($V_{dv,ri}$, $V_{dw,ri}$), ($V_{dv,rj}$, $V_{dw,rj}$), …, ($V_{dv,rk}$, $V_{dw,rk}$) are all couples of partially common values)» from definition 7, is the couple of tuples (e_{d3}, e_{d6}), e_{d3}=<$V_{d3,r2}$, $V_{d3,r3}$, $V_{d3,r4}$>, e_{d6}=<$V_{d6,r2}$, $V_{d6,r3}$, $V_{d6,r4}$>. No other couple of tuples satisfies the above condition. Therefore, now we only have to verify the second condition of definition 7, «(f(d_v)≠f(d_w)». From the OAT it follows that f(d3)={C_b)} and f(d6) ={C_b}, hence f(d3)=f(d6). Thus, it is concluded that R_x={r2, r3, r4} is an attribute basis.

In this subsection we have described the basic concepts to do with the second gen- eral step of the knowledge acquisition device: «reasoning about the resulting set of antecedent-consequent rules, in order to modify the KB, which adopts a decision tree format». This step involves, first, finding an attribute basis that satisfies the given criterion, which depends on the nature of the problem at hand. Second, generate a decision tree from the attribute basis found. However, before proceeding, it is neces- sary to turn all the input knowledge to the KAD - that is, the examples from the envi ronment and the knowledge base - into the same format. The following subsection describes the basic concepts to do with such a conversion process.

OBJECT-ATTRIBUTE TABLES AS SETS OF RULES

This section describes the general aspects of the process to turn a knowledge base expressed in a decision tree format into a set of condition-action rules.

Condition-action rules -also named production rules- are one of the most common ways to express the knowledge base of a reactive agent [9, 10]. The antecedent of the rules is made up of conjunctions of attribute-value pairs, while the consequent con- sists of a single action - also called concept - which can be triggered under the as- sumption of the truth of its antecedent. A rule base is a set of rules.

Decision trees constitute an intensive -abstract- format to describe an OAT. Each inner node of a decision tree -also called attribute-node- generated from an OAT rep- resents an attribute, whereas each leaf node -also called concept-node- represents a subset of concepts. Each attribute-node has as many branches as values in the attrib- ute domain associated with the node. A decision tree covers all the examples of the corresponding OAT.

Given a decision tree, each branch from the root of the tree to a leaf node repre- sents a rule. Thus, a decision tree can be represented as a set of rules. Figure 2 illus- trates an example.

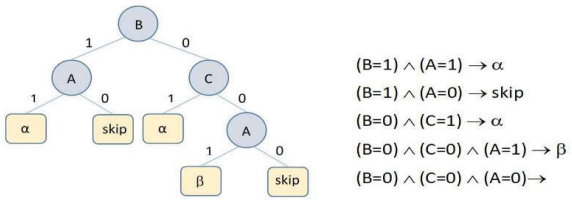

Fig. 3. Decision-Tree as a set of rules.

{A, B, C} is the set of attributes of the decision tree in Figure 3 and {α, β, skip} is the set of concepts.

The following definition provides the characteristics that a base of production rules must meet in order to be suitable for applying an inductive inference procedure to it.

Definition 9 (full rules). Let AT be the set of attributes through which the anteced- ents of all the rules in a production rule base, BC, are described. A rule whose ante- cedent include all the attributes in the set AT is called a full rule.

For example, rules (B=1)^(A=1) → α and (B=1)^(A=0) → skip in Figure 3 are not full rules, since their antecedent does not contain all the attributes of AT={A,B,C}. While rule (B=0)^(C=0)^(A=1)→ β is a full rule.

Let V_i be the domain of values of the attribute r_i in a rule base. If a rule is not a full rule, then it can be turned into a full rule. To do this, it is necessary to complete its antecedent by adding to it those attribute-value pairs whose attributes, $r_i \in AT$, do not appear in the antecedent of the rule and whose values of said attributes will corre- spond to the value $*^{i}_{j,k,\ldots p,}$, where {j, k,..., p}=V_i is the domain of values of r_i. For example, from Figure 3 we have that $V_A=V_B=V_C=\{0,1\}$. For example, rule (B=1)^(A=1)→α is not a full rule, but it can be turned into the following full rule: (B=1)^(A=1)^(C= C0,1)→α. Figure 4 shows the OAT from Figure 3 where all its rules have been turned into full rules.

$(B=1) \wedge (A=1) \rightarrow \alpha$	$(B=1) \wedge (A=1) \wedge (C=*^{C}_{0,1}) \rightarrow \alpha$
$(B=1) \wedge (A=0) \rightarrow$ skip	$(B=1) \wedge (A=0) \wedge (C=*^{C}_{0,1}) \rightarrow$ skip
$(B=0) \wedge (C=1) \rightarrow \alpha$	$(B=0) \wedge (C=1) \wedge (A=*^{A}_{0,1}) \rightarrow \alpha$
$(B=0) \wedge (C=0) \wedge (A=1) \rightarrow \beta$	$(B=0) \wedge (C=0) \wedge (A=1) \rightarrow \beta$
$(B=0) \wedge (C=0) \wedge (A=0) \rightarrow$ skip	$(B=0) \wedge (C=0) \wedge (A=0) \rightarrow$ skip

Fig. 4. Complete rules of an OAT.

An OAT can be considered as a set of full rules. Those OATs resulting from the process of turning not full rules into full rules are, in general, incompletely specified OATs and can be handled by the corresponding inference mechanisms [3, 10].

Definition 10 (OAT of rules). An OAT of rules is defined as a structure made up of production rules such that all rules are full rules.

From now on all considered OATs will refer to OATs of rules.

INDUCTIVE INFERENCE ON OATS OF RULES

The question at this point is «how to reason about an OAT expressed as a set of rules», that is, an OAT of rules, «another set of rules expressed as a decision tree».

The most common inference from a set of rules is deductive inference, character- ized by the search for specific conclusions from assumptions represented through general knowledge. Inductive inference, on the other hand, represents reasoning on specific elements of knowledge in order to obtain general conclusions. Thus, it seems that none of the types of deductive and inductive inference fit the task actually posed at this point; that is, a general knowledge structure - decision tree - must be obtained from another general knowledge structure - rule base. There is little literature about the proposed problem. With the exception of [10], all other references that have been considered are based on points of view very different from those of this work.

The algorithm proposed in [10] allows obtaining a decision tree from an OAT of rules with priorities defined on the rules. The algorithm presented in [2] constitutes the central core of the KAD. This algorithm constitutes an extension of that presented in [10], so that it allows reasoning jointly from an OAT of examples coming from the environment, an OAT of rules coming from the decision tree - which constitutes the knowledge base, KB - and the feedback, in order to improve the decision tree -that is, the knowledge base-.

This algorithm works cyclically, so that each cycle constitutes a learning cycle. The number of learning cycles of the algorithm is unlimited, unless the environment is no longer able to provide new examples and, at the same time, the execution- verification device is also not capable of generating new experiences. As long as only one or both

devices - the environment or the execution-verification device - is still capable of providing new knowledge - examples/experiences - then the algorithm will continue working, since the knowledge base can still be improved. Thus, the algo- rithm is limited when both the capacities of the environment and those of the execu- tion-verification device are exhausted at the same time.

A detailed analysis of the learning algorithm described in [2] makes it possible to highlight some important properties:

- The algorithm implements a dynamic learning process, which allows, in each learning cycle, to update the current knowledge base based on the generation of new experiences and/or when new examples appear from the environment.

- The learning process of the described model is carried out under the closed domain hypothesis. In this way, the evolution of the knowledge base is based exclusively on the knowledge provided by all the examples of the environment that have ap- peared up to the current moment. In this way, the learning process updates, cycle by cycle, the knowledge base.

- The sets of attributes and their domains are not necessarily the same in each learning cycle. That is, in a learning cycle new attributes that have not appeared in pre- vious cycles may appear. Similarly, attributes that have appeared in previous cy- cles may not appear in a learning cycle. This fact opens the door to more powerful inductive learning systems.

- Each learning cycle of the algorithm results in a necessarily non-negative evolution of the knowledge base. The term "non-negative" is used because there is a possibil- ity that the new examples or experiences that appear in a given cycle do not pro- vide sufficient qualitative knowledge to allow the knowledge base to be improved. However, the knowledge base can never get worse in any of the cycles. This is a natural form of learning.

- The possibility of performing inductive inference on incompletely specified exam- ples is one of the great contributions to learning, and it also facilitates some of the previous properties.

- The activation of a new learning cycle occurs for three reasons: simply due to the appearance of new examples - as described above -; simply by the emergence of new experiences provided by the Action-Decision device; or for both of the above reasons.

- It is possible to provide the learning algorithm with discretization processes of continuous attributes [4] and binarization processes of discrete attributes [17].

- The learning algorithm is clear, simple and well structured, which makes it easy to implement in a programming language.

- This algorithm is not limited in time.

- Some applications of inductive inference in different fields, such as medicine, stock market analysis, disability assistance, and web page classification, are de- scribed in [11, 12, 13, 14, 15, 16]. These applications were developed using the UIB-IK inductive platform [8], designed for completely specified OATs, including data of different typologies.

The following section describes the most relevant features of the hypothesis or knowledge base.

THE HYPOTHESIS OR KNOWLEDGE BASE

The knowledge base adopts a decision tree format that evolves over time. Further- more, it must be taken into account that the learning process is carried out under the fundamental hypothesis of the closed domain. That is, the learning cycle at the current time must consider only the characteristics of all the examples that have appeared since the first learning cycle. This means that those examples that have not yet ap- peared in any learning cycle must not be considered in the current decision tree. The following subsections will help us to understand how evolutionary inductive learning can be handled.

DECISION TREES FOR LEARNING

To update the evolutionary hypothesis/ decision tree in the current learning cycle, it is necessary, first of all, to know which examples from the environment have appeared and which have not appeared -missing examples- in previous learning cycles. To do this, a kind of decision trees known as *decision trees for learning* – abbreviated DTL- will be used. Once the decision tree for learning has been generated in the current cycle, the *final decision tree* –abbreviated FDT- is then generated, which exclusively covers examples that have appeared from the first cycle to the current cycle.

As an illustrative example of the characteristics of a DTL, without going into deeper issues, consider the OAT of rules with three rules in Figure 5.

$$(A=0) \wedge (B=0) \wedge (C=0) \rightarrow C1$$
$$(A=0) \wedge (B=1) \wedge (C=0) \rightarrow C2$$
$$(A=1) \wedge (B=0) \wedge (C=1) \rightarrow C3$$

Fig. 5. OAT of rules.

Note how the domains of attributes A, B, C in Figure 5 are binary.

First, a decision tree for learning -DTL- inferred from the examples in Figure 5 is generated. This decision tree covers, on the one hand, these examples and, on the other hand, the rest of the possible examples of the domain that do not appear in Fig- ure 5. Such not appeared examples are denoted by the symbol *n.a* -abbreviation of not appeared. Figure 6 shows an example of such a decision tree for learning.

Once the decision tree for learning has been generated in the current cycle, the *final decision tree* is then generated, which exclusively covers examples that have appeared from the first cycle to the current cycle.

Figure 7 shows the final decision tree for the current cycle. This tree satisfies the minimum number of nodes criterion.

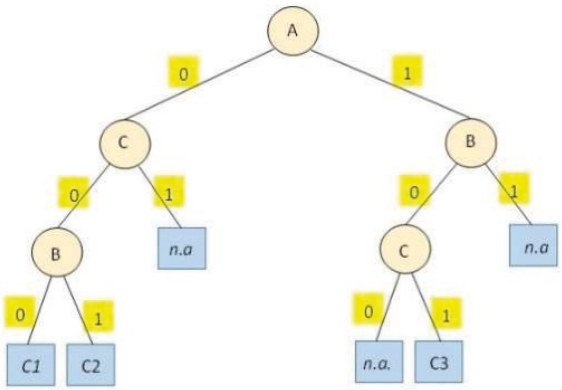

Fig. 6. Decision tree for learning –DTL.

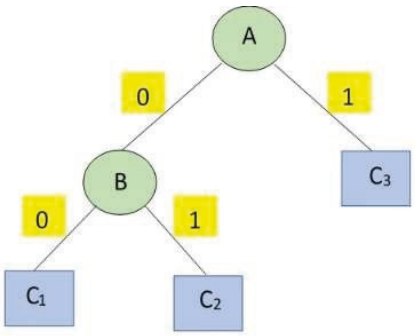

Fig. 7. Final decision tree –FDT.

LEARNING DECISION TREES

For ease of discussion, only the Feedback 1 Model 1 will be considered at this point. The inferential process in charge of updating the DTL in each learning cycle of a Feedback-1 model is based on the following two knowledge elements:

- The new examples of the OAT of rules that appear in the current learning cycle. These are the new examples coming from the environment in this current learning cycle.
- The current DTL, which corresponds to the DTL generated in the preceding learn- ing cycle.

Note that each of the two previously mentioned elements of knowledge are described through an OAT of rules. Both OATs are then unified into a single OAT of rules, which requires some specific processes that are not described in this work.

In [2] is described algorithm that, on the one hand, unifies both OATs of rules and, on the other hand, updates the current DTL from the resulting unified OAT. Some important properties of this algorithm are highlighted in section 4.3.

CONSIDERATIONS REGARDING THE FEEDBACK 2 MODEL

Known as experience-driven learning, Feedback_2 is a feedback model whose function is to update the current Knowledge Base. To do this, it is based on the experienc- es generated by the Action-Decision Device and, possibly, the appearance of new examples from the Environment.

EXPERIENCES AS A CONSEQUENCE OF DECISION MAKING

Some general issues related to generating experiences are:

- When and how should experiences be generated?
- Who is responsible for generating them?
- What are the reasons that lead to the need for experiences?
- What kinds of experiences should be considered?
- Do experiences depend on the nature of each particular problem or do they consti- tute generic patterns for an entire class of problems?
- Is it possible to automate any type of experiences and thus obtain a completely autonomous learning system?
- How to turn the different types of experiences into formal expressions so that they are acceptable to the knowledge acquisition device?

This section proposes a general approach that addresses some of the issues mentioned above.

Looking at the model in Figure 1, the responsibility for generating experiences lies with the *execution-verification device*. Whatever the case, these decisions/actions must first be carried out on the agent's environment and then, they must be evaluated for their success or failure. This evaluation is performed by the *verification device*. Thus, the model in Figure 1 can be extended by that in Figure 8.

The type of experiences of a learning system depends on the nature of the problem for which it was designed. Thus, a single learning system can incorporate experiences of different kinds and sources, such as:

- Experiences guided by the user/expert of the system, who may or may not feel satisfied with the decisions made by the learning system. For example, the user de- cides that the criterion defining the characteristics that the Hypothesis – Decision Tree– must satisfy, must be modified.

- Experiences generated by the system itself, which can detect certain situations that demand consideration. For example, the system detects that the decision on some examples in the Fact Base is not satisfactory.

- Experiences due to changes in the characteristics of the environment. For example, some attributes become unimportant due to disturbances in the environment. This suggests that the attribute basis on which the Hypothesis is based is unsatisfactory and needs to be modified.

- Etc…

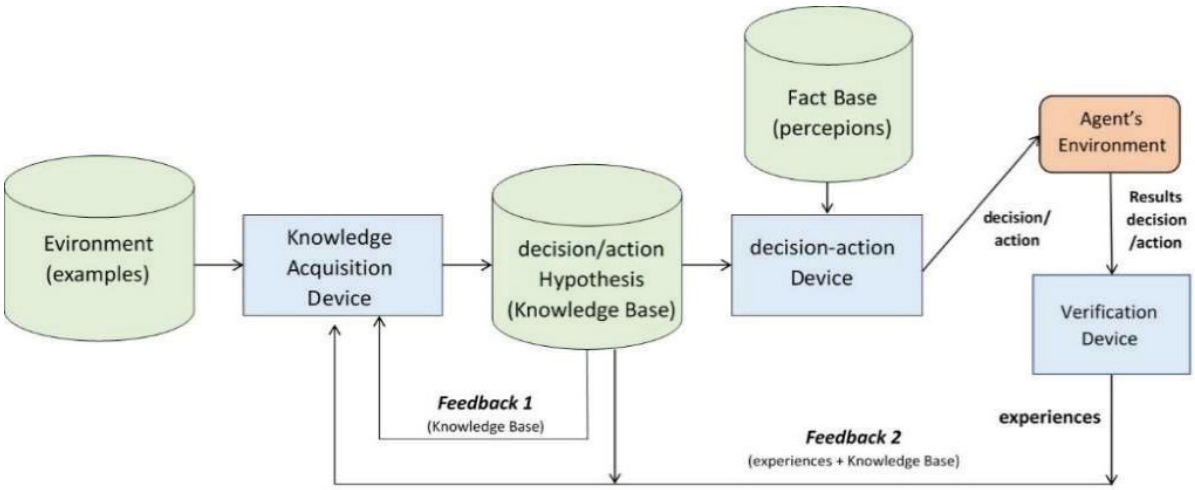

Fig. 8. Autonomous inductive learning model extended with experiences.

In any case, the concept «experiences» can be too complex and excessively broad to be addressed in a unified way. In [2] a set of possible experiences for inductive learn- ing systems from examples is presented.

REFERENCES

1. Cohen, D. R., Feigenbaum, E, A.: The Handbook of Artificial Intelligence, vol 3. William Kaufman, Inc., Stanford, Calif. (1982).

2. Fiol Roig, G.: Autonomous Learning from Examples: An Incremental Inductive-Deductive Model (English Edition). Our Knowledge Publishing, Chisinau (2023).

3. Fiol Roig, G.: Inductive Learning from Incompletely Specified Examples. Frontiers in Ar- tificial Intelligence and Applications 100, 286–295 (2003).

4. Fiol Roig, G.: Learning from Incompletely Specified Object Attribute Tables with Contin- uous Attributes. Frontiers in Artificial Intelligence and Applications 113, 145–152 (2004).

5. Fiol Roig, G., Miró Nicolau, J., Miró Julià, J.: A New Perspective in Inductive Acquisition of Knowledge from Examples. Lecture Notes in Computer Science 682, 219-228 (1993).

6. Fiol Roig, G.: On Qualitative Knowledge in a Rule Based Knowledge Bases. Proceedings of the IJCAI-93 workshop on Validation, Verification and Test of KBSs, pp. 27-36. Chambery (1993).

7. Pawlak, Z.: Theoretical Aspects of Reasoning About Data. Warsaw University of Tech- nology. Institute of Computer Science, 1990.

8. Fiol Roig, G.: UIB-IK: A Computer System for Decision Trees Induction. Lecture Notes in Artificial Intelligence 1609, 601-611 (1999).

9. Russell, S., Norvik, P.: Inteligencia Artificial. Un enfoque moderno, 2nd edn. Pearson Educación S.A., Spain (2004).

10. Fiol Roig, G.: On Discovering Qualitative Knowledge in Rule Based Knowledge Bases. An Intelligent Approach. Proceedings of CISTI'2020 - 15th Iberian Conference on Infor- mation Systems and Technologies. Seville (2020).

11. Fiol Roig, G., Miró Nicolau, J.: A Diagnosis Problem Approach based on Inductive Ac- quisition of Knowledge from Examples. Heuristics Vol 6(3), 54-65, (1993).

12. Fiol Roig, G. et al.: Computer-Aided Causal Diagnosis of Ascites. Analysis of a Prototype. Information, Intelligence and Systems Vol. 2, 1102-1107, (1996).

13. Fiol Roig, G., Arellano, D., Perales, F.J., Bassa, P., Zanlongo, M.: The Intelligent Butler: A Virtual Agent for Disabled and Elderly People Assistance. Advances in Soft Computing Vol. 50, 375-384, (2008).

14. Fiol Roig, G., Miró Julià, M.: A Contribution for Elderly and Disabled Care Using Intelli- gent Approaches, Lecture Notes in Computer Science 5518, 902-905 (2009).

15. Fiol Roig, G., Miró Julià, M.: Stock Market Analysis using Data Mining Techniques: a Practical Application. International Journal of Artificial Intelligence Vol. 6 (11), 129-143 (2011).

16. Fiol Roig, G., Miró Julià, M., Herraiz, E.: Data Mining Techniques for Web Page Classifi- cation. Advances in Intelligent and Soft Computing Vol. 89, 61-68, (2011).

17. Fiol Roig, G.: Contribución a la Adquisición Inductiva de Conocimiento. Doctoral disser- tation. University of the Balearic Islands, Palma (1991).