ELSEVIER

Fuzzy Sets and Systems 123 (2001) 307–320

FUZZY sets and systems

www.elsevier.com/locate/fss

Use of a fuzzy machine learning technique in the knowledge acquisition process

J.L. Castro[a], J.J. Castro-Schez[b], J.M. Zurita[a, *]

[a] *Departamento Ciencias de la Computación e IA, ETSI Informática, Universidad de Granada, Avenida de Andalucía, 38, 18071 Granada, Spain*

[b] *Departamento de Informática, EU Informática, Universidad de Castilla-La Mancha, Ronda de Calatrava, 5, 13071 Ciudad Real, Spain*

Received 22 July 1998; received in revised form 27 June 2000; accepted 8 November 2000

Abstract

Acquiring the knowledge to support an expert system is one of the key activities in knowledge engineering. Knowledge acquisition (KA) is closely related to research in the machine learning field. Any machine learning acquires some knowledge, but not enough knowledge for building expert systems. The aim of this article is to present a new approach to machine learning which helps to acquire knowledge when building expert systems. This technique will acquire the more general knowledge that should be used for extending, updating and improving an incomplete and partially incorrect knowledge base (KB). The main claim of our approach is that the system will start with poor knowledge, provided by the expert or the organization to which he belongs. A machine learning technique will evolve it to an incomplete KB, which may be used for further interactions with the expert, that will incrementally extend and improve it until obtaining a complete KB (i.e., with complete inferential capabilities).

Keywords: Knowledge acquisition; Machine learning; Genetic algorithms

1. Introduction

An important part of building expert systems is acquiring the knowledge needed to achieve the desired levels of performance. Most of the time required for the development of expert systems can be attributed to knowledge acquisition. When a knowledge engineer faces this problem, the main difficulty will be the expert's inaccessibility. The organization does not want its expert to lose his time. In most cases, the organization will require that the knowledge engineer be familiar with the structure and representations of the system before interviewing the domain expert. The knowledge engineer will have a degree of mastery over the domain and a sense of the kind of knowledge that will result in better performance. Hence, the time lost by the expert is lower.

Sometimes, the organization will provide the knowledge engineer with a set of the expert's past actions, which shows the way the expert reasons.

* Corresponding author. Tel.: +34-958-244019; fax: +34-958-243317.

E-mail addresses: castro@decsai.ugr.es (J.L. Castro), jjcastro@inf-cr.uclm.es (J.J. Castro-Schez), zurita@decsai.ugr.es (J.M. Zurita).

0165-0114/01/$ - see front matter
PII: S0165-0114(01)00008-2

The knowledge engineer acquires knowledge learning either by observation or by using examples. The knowledge engineer analyses the set of past actions, he observes examples of commonly occurring problems, as well as "hard cases" and notes the expert's responses in each case. The expert does not explicitly instruct the knowledge engineer in all of the heuristics that are applied during the problem-solving activities, but the engineer begins to learn by *induction*. Induction is the process that allows us to infer a "law" or develop theories by observing and analysing events or cases in which that law is applied. Thanks to the induction process we attain general conclusions from specific examples. After this, the knowledge engineer will be familiar with the domain and will perform interesting interviews.

Induction has become an effective method for knowledge acquisition in those cases in which there is incomplete knowledge, the expert is not accessible or the expert is not able to totally explain his or her way of performing, that is, the processes or heuristics which lead to solving the problem. Experts have a natural tendency to provide examples of problems and sample solutions. Thus, it is reasonable to think about automating the induction process if we want to automate the knowledge acquisition process. Induction-based knowledge acquisition requires that the domain expert or the organization to which he belongs, provide the examples and critical information by entering them in a computer program, which builds an induction table for data analysis. Knowledge engineers can acquire important domain knowledge using this approach.

Machine learning research has developed methods that are interesting and very important. Successful applications of ID3 and C4 [15] and similar ones have provided knowledge bases for working expert systems whose task is classification. In some cases, the rules generated for those inductive learning methods [3–6,9,14] are more effective than the rules that were generated manually by a domain expert. We mentioned earlier that some machine learning techniques are useful in knowledge acquisition but the mere fact that we can infer theories from examples does not guarantee the acquisition of high-quality knowledge.

So machine learning techniques cannot be treated as completely automated knowledge acquisition tools, though they do acquire some knowledge. Machine learning techniques may help in avoiding some difficulties, which are always present in the knowledge acquisition process, such as the expert's inaccessibility, difficulties in reporting knowledge. Moreover, these techniques allow us to take advantage of some benefits which are manifest in the examples, such as extracting and translating evident expert knowledge or heuristics, which are present in the examples, into rules.

The main objective of the present paper is to describe a new machine learning technique that can help in acquiring knowledge when building expert systems. We do not mean that the newly developed machine learning technique will be complete; our idea is to obtain a technique to take advantage of the benefits of examples. The main claim of our approach is that the system will start with poor knowledge, generally a set of examples that shows the way the expert reasons, provided by the expert or the organization to which he belongs. A machine learning technique will evolve it to an incomplete KB, a set of initial rules, which should be used for further interactions with the expert, that incrementally will extend and improve it until obtaining a complete KB, with complete inferential capabilities. The rules generated by the new technique should have a maximal structure in the sense of both having less components in the antecedent part of the rule and simultaneously identifying the largest number of examples in the given set of input–output data. The algorithm will generate a minimal set of maximal rules to identify the evident knowledge inherent to the set of examples.

These generated rules ought to be used for developing a set of interview strategies that appeared to rapidly lead to a more powerful rule base. These strategies will provide a mechanism for interviewing domain experts and would not only relieve the knowledge engineer of the interview burden, but would also allow a rapid assessment of the strength of the current knowledge base and a selection of interview questions designed to compensate for these weaknesses. Tecuci [18] has also proposed this approach.

This paper is organized as follows. Section 2 presents the general ideas on which our technique is based. In Sections 3 and 4, we present the technique developed for knowledge acquisition, which will acquire only the evident knowledge. Section 5 shows the experimental results obtained with the technique. Section 6 outlines the way we could make use of the

results of our technique for obtaining a better knowledge base. The final section outlines some directions for future research.

2. General ideas of knowledge representation

For most real-world problems, the information concerning design, evaluation, realization, etc., can be classified into two kinds: numerical information (a set of examples) obtained from observation and linguistic information obtained from human experts. The experience of the expert is usually expressed as some linguistic "IF-THEN" rules that state in what situation which action should be taken. The set of examples gives the specific values of the inputs and the corresponding successful outputs.

We start out from a poor knowledge base, generally a set of numerical examples that shows the way the expert reasons, provided by the expert or the organization to which he belongs, and we shall attempt to evolve it into an incomplete knowledge base which will be employed for further interactions with the expert, that will incrementally extend and improve it until a complete KB is obtained. The knowledge of the incomplete KB obtained for us must be represented in such a way that it will be understood by the expert, with the aim of allowing the interactions mentioned earlier.

In the last few years, research in the knowledge representation field has led to the fuzzy logic field. Zadeh [20] suggested a device for modelling human behaviour, which is based on fuzzy linguistic values.

At the present time, there are a great number of knowledge-based systems (KBS) developed making use of fuzzy logic [10,13]. These systems make use of fuzzy logic as a mechanism for knowledge representation because it allows domain knowledge to be expressed in a manner more similar to how it is represented in the expert's mind. The systems are described by means of a set of IF-THEN rules with vague predicates.

These types of systems have led to the research of tools for designing KBS [11,12] as well as the appearance of a large number of papers devoted to the study of these systems. Among these papers there are many devoted to inductive learning. The idea of many inductive learning methods is to extract a set of fuzzy rules from a set of examples, which show the way the system works. Each author uses a different methodology. Thus, we can find methodologies such as: fuzzy clustering [19], neural networks [2,16], utilization of an assumption-based truth maintenance system (ATMS) [5], definition and use of activation hyperboxes over the domain variables [1].

We are going to develop an inductive learning method that generates a minimal set of fuzzy IF-THEN rules capable of modelling the evident expert knowledge inherent to the set of examples.

We start out from a set of examples that shows the way the expert reasons:

$$\theta = \{e_1, \ldots, e_m\},$$

$$e_i = ((x_{i0}, \ldots, x_{in}), y_j),$$

where m is the number of reasoning examples, n is the number of variables used in the reasoning process, y_j is the output value, and $x_{i0}, \ldots, x_{in}$ are the values of the variables used in the reasoning process (input variables) and y_j is the value of the expert's response (output variable). This set is a subset of the product of the inputs and outputs ($X^n \times Y$). Our first objective will be to approximate the function: $\Omega : X^n \to Y$, that completely models the way the expert reasons.

That function will be approximated by means of a set of fuzzy rules, which are obtained from the set of examples. This set of fuzzy IF-THEN rules will identify the evident knowledge present in the set of examples that shows the way the expert reasons.

The form of our fuzzy rules will be the following:

R_j: IF X is E THEN Y is y_j,

where X is the set of variables used to reason, E is a set of subsets of fuzzy labels and Y is the expert's response. Another way to write the rule is

IF X_0 is E_0 and X_1 is E_1 and ... and X_n is E_n

THEN Y is y_j,

where X is E, is a contraction of the expression X_0 is E_0 and X_1 is E_1 and ... and X_n is E_n, with X being the set $(X_0, X_1, \ldots, X_n)$ of input variables and E the set $(E_0, E_1, \ldots, E_n)$ of sets of labels.

We shall assume that the fuzzy sets E_i (values of X_i in the rule) are taken from the set of labels $\mathscr{L}_i$, with $\mathscr{L}_i$ being $\{L_{i1}, \ldots, L_{ik}\}$ where k indicates the number of

labels. Therefore $E_i \in \mathscr{P}(\mathscr{L}_i)$, with $\mathscr{P}$ being the function of parts of a set.

3. Learning evident rules from a set of examples

In this paper, we propose a procedure for generating fuzzy rules which models the expert's evident conduct from a set of examples that shows the way the expert reasons. For this purpose, we suggest using the algorithm developed by Castro et al. [3], which generates a set of fuzzy rules that identify a system. The rules generated for this algorithm will have a maximal structure in the sense of both having less components in the antecedent part of the rule and identifying the largest number of examples in the given set of input–output data. This makes it interesting for our technique of knowledge acquisition. However, this algorithm has one problem in that its purpose is not knowledge acquisition, but the identification of system performance. This problem is the quantity and generality of the rules generated.

The aim of this paper is to reduce the number of rules in order to obtain a set of more general rules, which allow interactions with the expert that will incrementally extend and improve it until obtaining a complete KB. The idea is as follows: Constructing a set of initial rules which have a maximal structure and permitting their evolution making use of a "*genetic algorithm*" until we obtain the set of better rules.

The set of initial rules will be obtained from the set of training examples, making use of the algorithm proposed by Castro et al. [3]. Let us remind ourselves about the ideas in that algorithm:

The key idea of the algorithm is as follows: Constructing an initial set of fuzzy rules from a set of examples and then constructing a set of definitive or maximal rules which are generalizations of those initial rules.

It divides each input space i, where is defined the input variable X_i into fuzzy regions (each region will be assigned a fuzzy membership function μ_{ij}) and encodes the output. Then, it generates a set of initial fuzzy rules from given examples. To do so, the values of the input variables in each example will be represented by means of the label which presents the maximum degree (according to $L_{ij} = \max_j\{\mu_{ij}(x_i)\}$), and the output is encoded.

Example 3.1. Let $((a,b,c,d),z)$ be one example of the training set, where a,b,c and d are concrete values of the input variables X_0,X_1,X_2 and X_3 and z is the value of the output variable. First, determine the degrees of a,b,c and d in different regions. Second, assign a given a,b,c and d to the region with the maximum degree.

Let us assume that the maximum degrees are as follows: for X_0 it is $j=1$ $(\mu_{01}(a))$, for X_1 it is $j=3$ $(\mu_{13}(b))$, for X_2 it is $j=2$ $(\mu_{22}(c))$, and for X_3 it is $j=1$ $(\mu_{31}(d))$ then this example will be translated into the following rule:

IF X_0 is $\{L_{01}\}$ and X_1 is $\{L_{13}\}$ and X_2 is $\{L_{22}\}$ and

X_3 is $\{L_{31}\}$ THEN Y is y_i,

where L_{01},L_{13},L_{22}, and L_{31} are the labels that present the maximum degree and y_i represents the output.

At this point, the set of fuzzy rules that identify the system will be constructed. To do so, it takes each initial rule and it checks whether that rule subsumes in some rule of the set of definitive rules. At the beginning this set will be the void set.

We should recall that one rule R_i $((E_{i1},E_{i2},\ldots,E_{in}),$ $y_j)$ subsumes into another rule R_k $((E_{k1},E_{k2},\ldots,E_{kn}),$ $y_x)$, that will be noted as subsume (R_i,R_k) if $E_{i1}\subseteq E_{k1},E_{i2}\subseteq E_{k2},\ldots,E_{in}\subseteq E_{kn}$ and $y_j = y_x$.

Example 3.2. The rule $((\mathrm{SN},\mathrm{HP},\mathrm{HN},\mathrm{HN}),y_1)$ subsumes into the rule: IF X_3 is $\{\mathrm{HN},\mathrm{MN}\}$ THEN Y is y_1. When an input variable does not appear in the rule, this means that it can take any value from the set of the labels in this variable. In this way, the set of labels is $\mathscr{L} = \{\mathrm{HN},\mathrm{MN},\mathrm{SN},Z,\mathrm{SP},\mathrm{MP},\mathrm{HP}\}$ for the variables X_0, X_1 and X_2. The rule $((\mathrm{SN},\mathrm{HP},\mathrm{HN},\mathrm{HN}),y_1)$ subsumes into the rule $((\mathscr{L},\mathscr{L},\mathscr{L},\{\mathrm{HN},\mathrm{MN}\}),y_1)$, obviously $\mathrm{SN}\subseteq\mathscr{L}$, $\mathrm{HP}\subseteq\mathscr{L}$, $\mathrm{HN}\subseteq\mathscr{L}$, $\{\mathrm{HN}\}\subseteq\{\mathrm{HN},\mathrm{MN}\}$ and $y_1 = y_1$.

If the rule subsumes into some definitive rule, then it is ignored, since that rule is already generalized, and it takes a new rule from the set of initial rules. However, if the rule does not subsume into any definitive rule, the algorithm works with that initial rule applying a process of amplification. This process consists of the following:

It starts out from that initial rule $((E_1, E_2, \ldots, E_n), y_j)$, and it goes on performing amplifications in each variable, i.e., it amplifies each E_i, $i = 1, \ldots, n$ with the labels that have still not been considered for it.

Example 3.3. If the set of labels is $\mathscr{L} = \{N, Z, P\}$ and we have the rule $((N, Z, P), y_3)$, then the amplifications are carried out in the following way:

$$((N,Z,P), y_3) \to ((\{N,Z\}, Z, P), y_3)$$

$$\to ((\{N,Z,P\}, Z, P), y_3) \to$$

$$((\{N,Z,P\}, \{Z,N\}, P), y_3)$$

$$\to ((\{N,Z,P\}, \{Z,N,P\}, P), y_3) \to$$

$$((\{N,Z,P\}, \{Z,N,P\}, \{P,N\}), y_3)$$

$$\to ((\{N,Z,P\}, \{Z,N,P\}, \{P,N,Z\}), y_3).$$

When we try to carry out amplification, it checks whether it is possible.

An amplification from R_i to $R_{i'}$ (with $R_{i'}$ being the rule: IF X is $E_{i'}$ THEN y_p), will be *possible* if there is no rule R_j (with R_j being the rule: IF X is E_j THEN y_q) in the set of initial rules that will verify that $E_j \subseteq E_{i'}$ and will not subsume $(R_j, R_{i'})$, that is, there is no rule R_j which will verify that $E_j \subseteq E_{i'}$ and $y_p \neq y_q$.

Example 3.4. If the set of labels is $\mathscr{L} = \{N, Z, P\}$ and we have the rule $((N, N, Z), y_2)$ in the set of initial rules, then an amplification from $((\{N,Z,P\}, Z, Z), y_3)$ to $((\{N,Z,P\}, \{Z,N\}, Z), y_3)$ is not possible, since $N \subseteq \{N,Z,P\}$, $N \subseteq \{Z,N\}$ and $Z \subseteq Z$ and $y_2 \neq y_3$.

When it carries out amplification from R_i to $R_{i'}$, if it is not possible then it continues working with R_i, if it is possible then it continues working with $R_{i'}$ after carrying out that amplification.

When the process of amplification concludes, the resulting rule will be included in the set of definitive rules.

For obtaining a smaller set of more general rules, we propose two improvements to this algorithm: ordering the input variables in the training examples, the most salient features will be located in the last positions and removing the noise present in the set of training examples.

3.1. Ordering the input variables

The philosophy of the algorithm proposed in [3] is to generalize each input variable of the example while it is possible. This leads to the fact that the last input variables in the example will be the most salient features.

The algorithm starts out from an initial rule $((E_1, E_2, \ldots, E_n), y_j)$ (an example of the set of training examples), and it continues performing amplifications in each variable, i.e., it amplifies each E_i, for $i = 1, \ldots, n$ with the labels that have still not been considered for it. When it carries out amplification R', it verifies if it is possible. For i with high values the possibility that one example e exists in the set of training examples (R is the set of initial rules) as $R \subseteq R'$ and does not subsume (R, R') is greater than for i with low values. This means that the amplifications for the last input variables will be more difficult.

We suggest that the input variables that are located in the last positions will be really the most salient features. To do so, we order the input variables in decreasing order with respect to their correlation with the output variable.

Our bottom-up method of induction, that goes from individual examples to global rules is improved by means of a top-down induction concept. When we examine the proposed algorithm in order to observe differences with other algorithms based on the top-down induction concept (such as decision tree learning, etc.), we conclude that the rules obtained by the method proposed are more legible, that is they are easier to read for an expert. Therefore, they can be used in later phases of knowledge acquisition process.

As an example we will see how we order the input variables in the Pima Indians Diabetes Database. Vincent Sigillito from Johns Hopkins University created this database. The data set contains 2 classes, Diabetes and No-Diabetes, and 768 instances (500 No-Diabetes; 268 Diabetes). The number of attributes is eight, all numeric valued: number of times pregnant, plasma glucose concentration at 2 h in an oral glucose tolerance test, diastolic blood pressure, triceps skin fold thickness, 2 h serum insulin, body mass index, diabetes pedigree function and age.

When we calculate the correlation of the input variables, we obtain the following results:

1. Plasma glucose concentration at 2 h in an oral glucose tolerance test (0.4666),

2. Body mass index (0.2927),
3. Age (0.2384),
4. Number of times pregnant (0.2219),
5. Diabetes pedigree function (0.1738),
6. 2 h serum insulin (0.1306),
7. Triceps skin fold thickness (0.0748),
8. Diastolic blood pressure (0.0651).

In this case the most salient feature is the *plasma glucose concentration at* 2 *h in an oral glucose tolerance test.* This is the variable that discriminates the most.

3.2. Removing the existing noise from the training set

We propose another improvement to this algorithm that allows us to obtain a smaller set of initial rules. Let us see what this improvement consists of.

We calculate, for each example in the set of training examples, two measures, which point out the distance with respect to the examples of the same type and with the examples of different types. These measures will be the following:

$$P_i^+(R_i) = \sum_j e^{-\alpha d(R_i,R_j)},$$

where R_j are examples of the same type.

$$P_i^-(R_i) = \sum_j e^{-\alpha d(R_i,R_j)},$$

where R_j are examples of different types.

In the P_i^- and P_i^+ calculation we make use of α-parameter, which will be used for penalizing the negative examples in the P_i^- calculation and the positive examples in the P_i^+ calculation. That is to say, negative examples raise the amount by which P_i^- increases. In a similar manner, positive examples raise the amount by which P_i^+ increases. In the test for the algorithm we have used $\alpha = 0.75$.

With d being a function which calculates the distance between two examples, for example, the *quadratic distance*:

$$d(R_i,R_j) = \sqrt{(e_i^1 - e_j^1)^2 + \cdots + (e_i^n - e_j^n)^2},$$

where $R_k = (e_k^1, \ldots, e_k^n)$.

We are assuming that each variable that describes examples is a numerical variable; however, the algorithm can be extended for working with other types of variables such as multipolar, boolean, graduated, fuzzy, etc. i.e., any type of knowledge used by the expert. This aspect will be studied in future works.

If an example has a value for $P^- \geqslant \varepsilon$, that is this example will be enclosed by many examples of different types, we can consider it to be a noisy example and we remove it from the set of training examples. We take ε as

$$\left(\left(\max_i\{P_i^-\} - \min_i\{P_i^-\}\right)\rho + \min_i\{P_i^-\}\right).$$

The value of ε will depend on examples from the set of training examples. It cannot be determined without knowing in what manner the examples are grouped. The value ρ establishes the P^- value of noisy examples that will be removed from the set. We have proven the algorithm with $\rho = 0.75$, that is we shall remove from the above-mentioned set those examples that have a value for

$$P^- \geqslant \left(\left(\max_i\{P_i^-\} - \min_i\{P_i^-\}\right)0.75 + \min_i\{P_i^-\}\right).$$

The set of examples obtained after removing noise will be set out in an increasing order with respect to its value of P^-, when two examples have the same value for P^- we will employ P^+ for ordering them. The example that has a higher value for P^+ will be located earlier. Thus the algorithm will generalize the least noisy examples first.

This improvement not only reduces the existing noise in the set of training examples and minimizes the amount of initial rules, but also helps to ensure that the knowledge acquired will be more evident knowledge. One drawback is that the algorithm will lose precision in the learning of the set mentioned earlier because we remove examples from the set that could not have any possible amplification. But it does not affect our aim of obtaining the evident knowledge present in that set.

4. Obtaining the best rules with the aid of a genetic algorithm

Once we have the initial rules, the idea of evolution comes into play, that is, permitting the evolution of this set of initial rules until obtaining a set of maximal or definitive rules which contains the more evident knowledge in the sense mentioned earlier.

To carry out this evolutive process we have considered the genetic algorithm, to which we have added some supplementary information (to manage some restrictions). The proposed algorithm is not exactly a genetic algorithm in the sense of Goldberg [8]. We do not attempt "to wade through" complicated spaces to search for optimal solutions. We are interested in consolidating a set of initial rules, thus we shall obtain the better rules or more evident rules present in the set of training examples.

Due to the cause mentioned earlier, we restrict the mutations and reproductions. We are not interested in jumps, which lead to new research spaces or eluding local solutions. Thus, we aim to obtain a minimal set of maximal rules to model the expert's evident knowledge.

4.1. Input specifications

The structure of the chromosomes of our individuals is a chain of elements which belong to the alphabet $A = \{0, 1, \#\}$. The length of the chain (a coded bitstring) depends on the number of fuzzy sets, which are used to divide the domain for each input variable and the length of the code, used for encoding the output. Hence:

$$\text{Length(chain)} = \sum_{i=0}^{n} |L_i| + \begin{cases} \text{Length(output code)} \\ \text{or} \\ |L_s|, \end{cases} \tag{1}$$

where $|L_i|$ is the number of fuzzy sets into which we have divided the domain of the variable X_i, $|L_s|$ is the number of fuzzy sets into which we have divided the domain of the output variable and Length(output code) is the length of the code used to encode the output value.

Example 3.5. If we have three input variables and their domains are divided into the following fuzzy sets: X_0 in five fuzzy sets, X_1 and X_2 in seven fuzzy sets and the output is encoded by means of a code whose length equals 1. The length of the chain will be as follows:

$$\text{Length(chain)} = 5 + 7 + 7 + 1 = 20.$$

The meaning of the alphabet elements is as follows: the 1 element is used for indicating the presence of the label in the rule. The # element indicates that this label may be present or not, it has not yet been decided. The 0 element states the absence of the label and its value must not be changed to 1.

As we can observe in formula (1) shown above, our chromosomes have two clearly distinguishable parts: a *kernel* that processes information about the individual and a *head* that processes information about the species to which it belongs.

The head is useful to satisfy one of our restrictions, which is that reproduction will only be possible between elements with the same head (individuals from the same species). A mutation in a bad place (or gene) can provoke perturbation about an example belonging to a given class. Thus, we consolidate the rules, which determine a class. This idea is not original since this is the way in which reproduction occurs in an ecosystem; there is reproduction only between elements of the same species.

Our objective will be to achieve a smaller number of rules which have a maximal structure in the sense of both having less components in the antecedent part of the rule and simultaneously identifying the largest number of examples in the given set of input–output data. Therefore, our rules will have to reproduce according to these two characteristics.

The set of initial rules will be obtained as was indicated at the beginning of Section 3. This data will be encoded as coded bitstrings.

Example 4.1. If we have three input variables and their domains are divided into the following fuzzy sets:

$$X_0 \to L_0 = \{N(g_1), Z(g_2), P(g_3)\}$$

$$X_1 \to L_1 = \{\text{HN}(g_4), \text{MN}(g_5), \text{SN}(g_6), \text{Z}(g_7), \text{SP}(g_8), \text{MP}(g_9), \text{HP}(g_{10})\}$$

$$X_2 \to L_2 = \{\text{HN}(g_{11}), \text{SN}(g_{12}), \text{Z}(g_{13}), \text{SP}(g_{14}), \text{HP}(g_{15})\}$$

and the output is encoded with a code whose length is equal to 1. According to this information, the chromosomes will have the form that is shown in Fig. 1.

If we obtain the rule IF X_0 is Z and X_1 is MP and X_2 is HN THEN Y is Iris Virginica as was indicated at

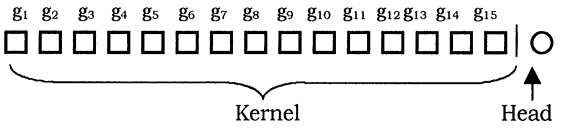

Fig. 1. Chromosome.

Fig. 2. IF X_0 is Z and X_1 is MP and X_2 is HN THEN Y is Iris Virginica.

the beginning of Section 3, this rule will be encoded as the coded bitstring shown in Fig. 2.

The rest of the values will be placed in #, to indicate that at the beginning it can take some value presence or absence.

The mutations in the genes, which distinguish between the species (head) will occur with a probability equal to zero, that is, the mutation of one species into another is totally impossible. Let us ask ourselves the following questions: Is it "really interesting" that the mutations are produced in our population when a mutation over an inadequate gene may lead to confusion over its belonging to one or another species? The answer is obvious: No. Though mutations are important in the evolutive process, guaranteeing no mutation for some genes is more or less coherent in our problem.

In the real world, there are some surveys, which show that mutations rarely occur, and that it depends on the species and genes. Hence, we can find that there are genes which change with relative facility (unstable genes), while others change very rarely (stable genes).

The chromosomes obtained from the initial population are pre-processed before performing the evolutive process. This pre-process consists of calculating the genes in which it is not convenient that mutations occur.

Let us look at how to find the genes on which mutations cannot be made.

Consider two different chromosomes, in the following sense, with different heads, and similar kernels which only have two different genes in the kernel with values # and 1 in one chromosome and values 1 and # in the other. In this situation, each non-specified value will change to value 0.

Fig. 3. Chromosomes in the initial population.

Fig. 4. Chromosomes after making non-mutable genes.

The mutation into some of these non-specified genes # provokes perturbation about this example belonging to one or the other class. This example will belong to different classes.

Example 4.2. If we have three input variables and their domains are divided into the following fuzzy sets:

$$X_0 \to L_0 = \{N, Z, P\}$$

$$X_1 \to L_1 = \{\mathrm{HN}, \mathrm{MN}, \mathrm{SN}, Z, \mathrm{SP}, \mathrm{MP}, \mathrm{HP}\}$$

$$X_2 \to L_2 = \{\mathrm{HN}, \mathrm{SN}, Z, \mathrm{SP}, \mathrm{HP}\}$$

and the output is encoded with a code whose length is equal to 1. Then we have in the initial population the individuals whose chromosomes are shown in Fig. 3.

As we can observe they differ in genes 9 and 10 of the kernel. We mark these genes as non-mutable genes and we obtain the individuals whose chromosomes are shown in Fig. 4.

Therefore, mutation in the example #1#######11 ####2 to #1######111####2 causes the example #1######1#1#### to subsume into two different classes: 2 and 3.

Over this set obtained from some non-mutable genes we apply the algorithm proposed in [3], and over the obtained rules for this algorithm we apply the suggested "*genetic algorithm*". The genetic algorithm will be applied to each class separately. Thus, reproduction will only be possible between elements with the same head (individuals of the same species).

If we have the rule IF X_1 is not $\{\mathrm{MP}, \underline{\mathrm{HP}}\}$ and X_2 is HP THEN Y is Iris Virginica in the set of definitive or maximal rules obtained by the algorithm developed in [3], this rule will be encoded as the coded bitstring

Fig. 5. IF X_1 not is $\{\mathrm{MP}, \mathrm{HP}\}$ and X_2 is HP THEN Y is Iris Virginica.

shown in Fig. 5, where not HP indicates that this gene is non-mutable.

The absence of the X_0 variable in the rule indicates that it can take any value. That the X_1 variable will be neither MP nor HP indicates that at the beginning it may be some value minus MP and it cannot be HP. However, in the evolutive process it may occur that the X_1 variable can take the MP value; the value HP cannot be taken because it is a non-mutable gene.

4.2. The suggested genetic algorithm

The genetic algorithm modifies the population of individuals in successive steps and will be applied to each class separately. The population in the state t is designated as $\alpha(t)$. The size of the population is kept constant at N individuals, where N is the number of definitive rules in the class to which we want to apply the genetic algorithm obtained by the algorithm [3]. The individuals, which will be reproduced, will be determined by how they adapt to the environment (the set of training examples). The adaptation of one individual to the environment will be decided by the fitness function. An appropriate fitness function for this problem is one that rewards the rules that classify the largest number of examples incurring the lowest number of errors. That is, a function $f_1(E)$ that will be increasing with respect to the number of correctly classified examples for $E(e_\mathrm{p})$ and decreasing with respect to the number of erroneously classified examples for $E(e_\mathrm{n})$.

$$f_1(E) = \frac{e_\mathrm{p}((e_\mathrm{tdc} - e_\mathrm{n})/e_\mathrm{tdc})}{e_\mathrm{tsc}},$$

where e_p is the number of examples in the training set (environment), which are correctly classified, e_n is the number of examples which are erroneously classified, e_tsc is the total number of examples from the same class that exist in the training set and e_tdc is the total number of examples from the different classes that exist in the training set.

It is also reasonable to reward $f_2(E)$ those individuals or chromosomes with a maximal structure, understood as those individuals or chromosomes with the largest number of specified genes (genes with values 0 or 1). Thus, we take the least complex rules.

$$f_2(E) = \frac{g_\mathrm{d}}{g_\mathrm{t}},$$

where g_d is the number of specified genes in the rule E and g_t is the number of genes that exist in a chromosome.

One possible fitness function will be the following:

$$f_\mathrm{fit}(E) = f_1(E) + f_2(E),$$

$$f_\mathrm{fit}(E) = \frac{e_\mathrm{p}((e_\mathrm{tdc} - e_\mathrm{n})/e_\mathrm{tdc})}{e_\mathrm{tsc}}\lambda + \frac{g_\mathrm{d}}{g_\mathrm{t}}(1-\lambda).$$

The symbol λ gives more significance to one or another aspect of the function. We have proven the algorithm with different values for λ. When λ value is near to value 1 we obtain a set of rules a little less general or maximal but which identifies the largest number of examples in the given set of input–output data. We have used $\lambda = 0.9$.

Definition 4.1. The individual E_i, whose chromosome is $e_1 e_2 \ldots e_n X$ is correctly classified by the rule R_j, whose chromosome is $r_1 r_2 \ldots r_n Y$, if the specified genes of the E_i (gene 1) verify that $e_k = r_k$ and both belong to the same species, that is $Y = X$.

Example 4.3. If we have the individual $E_x =$ 1#######1#1####3 and the rule $R_y = 101####$ 11011 11#3, then we can verify that this example E_i is correctly classified by the rule R_y, since $1(e_1) = 1(r_1)$, $1(e_9) = 1(r_9)$, $1(e_{11}) = 1(r_{11})$ and both heads are equal to 3.

Definition 4.2. The individual E_i, whose chromosome is $e_1 e_2 \ldots e_n X$ is erroneously classified by the rule R_j, whose chromosome is $r_1 r_2 \ldots r_n Y$, if the specified genes of the E_i (gene 1) verify that $e_k = r_k$ and they belong to different species, that is $Y \neq X$.

Example 4.4. If we have the individual $E_x =$ 1#######1#1####2 and the rule $R_y = 101####$ 1101111#3, then we can verify that this example E_i is erroneously classified by the rule R_y, since $1(e_1) = 1(r_1)$, $1(e_9) = 1(r_9)$, $1(e_{11}) = 1(r_{11})$ and their heads are different $3 \neq 2$.

Table 1
The crossover operator

∇	0	1	#
0	0	0	0
1	0	1	1
#	0	1	#

The steps in the genetic algorithm are as follows:

Step 1: Initialization. $\alpha(t)$ will be the initial population, which was obtained as explained earlier.

Step 2: Selection. For all i verify that $1 \leqslant i \leqslant N$, to reproduce the individuals in $\alpha(t)$ at one new population $\beta(t)$. The chromosome $E_i \in \alpha(t)$ appears in $\beta(t)$ with a probability equal to

$$p(E_i) = \frac{f_{\text{fit}}(E_i)}{\sum_{k=1}^{N} f_{\text{fit}}(E_k)}.$$

Step 3: Modification. Modify the chromosomes in $\beta(t)$ using the genetic operators to produce the new population $\alpha(t+1)$.

Step 4: Evolution. If the desired degree of evolution has not been obtained, do $t = t+1$ and go to Step 2.

Step 5: End.

We shall make use of the genetic operators for crossover and mutation. Let us look at how these operators are defined:

Crossover operator: The crossover operator is the most important genetic operator. Crossover combines, in some way, information on the two individuals who are reproduced over a new individual. The crossover operator that we shall use is ∇, which is defined as shown in Table 1.

Let A and B be two individuals from the same species and whose chromosomes are $a_1 a_2 \ldots a_n$ and $b_1 b_2 \ldots b_n$, respectively. The reproduction of these two individuals will be $a_1 \nabla b_1 \ldots a_n \nabla b_n$.

In the suggested learning procedure not all reproductions are permitted. Before reproducing two individuals, we verify that their breeding at least adapts itself to the environment and equals his or her parent's worst; if this does not occur then the reproduction will not be permitted. That is, crossover is possible when it verifies that

f_{fit} (breeding)

$\geqslant$ minimum (f_{fit} (father), f_{fit} (mother)).

A possible criticism that could be made of this restriction is the high computational cost, since we shall have to calculate the fitness function of pairs, which will be discarded. In fact, this cost is not too large because we admit reproductions only between individuals of the same species and therefore most of the time the reproduction improves on the parent (by the definition of the proposed crossover operator).

Mutation operator: The mutation operator will work over those genes not specified (#), the rest of the genes (0 or 1) are considered to be stable genes. The mutations of genes belonging to the head are not allowed because this could cause a change of species.

When the mutation arises, we verify whether it is malignant, that is the fitness function of the new individual deteriorates the value of the fitness function of the original individual. We say that a mutation is possible when it verifies that

f_{fit} (new individual) $\geqslant f_{\text{fit}}$ (original individual).

At this point, we may make the same remark about the computational cost. Why do we calculate the fitness function for mutations which will be rejected? This cost is compensated for since the introduction of noise (malignant mutations) in the population does destabilize the population, extra time will be necessary to get back to stability. Malignant mutations do not affect the problem of the search for a solution in a search space, and they can be presented as well since they can lead to new search spaces. Malignant mutations affect our problem, they increase the learning time because they interrupt stability.

4.3. The suggested algorithm for acquiring knowledge

We have developed an algorithm that takes a set of training examples as input and returns a minimal set of maximal rules. The basic idea of the algorithm is to acquire the evident rules from the set of examples that show the way the expert reasons.

This algorithm can be described in the following way:

Step 1: Order the input variables. We order the input variables; the most salient features will be located in the last positions.

Step 2: Remove the existing noise from the training set. We remove those examples that are "sufficiently"

far away from the remainder of the examples in the same class.

Step 3: Obtain non-mutable genes. We obtain some non-mutable genes, to prevent one mutation from being able to provoke disturbance about its belonging to one or another class.

Step 4: Obtain a set of initial rules. To do so, we suggest the algorithm proposed by Castro et al. [3].

Step 5: Apply the genetic algorithm to the individuals in each class. The algorithm will generate a minimal set of maximal rules which identify the evident knowledge in the set of examples.

Step 6: Remove unnecessary rules. We remove those rules included in other more general rules.

Step 7: End.

After applying the suggested algorithm, we obtain a set of rules with a maximal structure, which holds the evident knowledge present in the set of examples that show the way the expert reasons.

5. Experimental results

We have proven the algorithm with two databases, the *Iris Plants Database* and the *Pima Indians Diabetes Database*.

The Iris Plants Database was created by Fisher [7]. The data set contains three classes of 50 instances each, where each class refers to a type of iris plant. The number of instances in the Iris Plants Database is 150 (50 in each of the three classes), and the number of attributes is four numeric-predictive plus the class.

The first attribute indicates sepal length, the second sepal width, the third petal length and the last one indicates petal width. All the attributes are measured in centimetres. The class may be Iris-setosa, Iris-versicolor or Iris-virginica. There are values for all the attributes in the Iris Plants Database. In the test, we use 120 instances (80%) for learning and 30 instances (20%) for proving the rules that have been learnt.

Now, let us see how this algorithm reduces the number of rules obtained. The comparison will be made with the algorithm proposed in [3]. The results may be observed in Table 2.

The process of classification of one example is as follows:

We calculate the rules into which the example subsumes. The following two cases may arise:

Table 2
Percentage of success/number of rules

Test	Without reduction	With reduction
1	100/15	100/5
2	96.6/13	96.6/5
3	96.6/13	100/3
4	96.6/12	96.6/5
5	96.6/13	96.6/5
6	96.6/14	100/5
7	93.3/11	90/3
8	93.3/11	90/3
9	96.6/13	96.6/5
10	100/15	100/5

1. The example subsumes into one or more rules of the same class. In this case, our example is classified as an element from that class.
2. The example subsumes into two or more different classes of rules. In this case, we see the degree of convenience with each rule in which it is included and we classify the example as an element of the class with the greatest degree of convenience.

This algorithm helps to ensure that the knowledge acquired will be more evident knowledge. If not, let us see the rules obtained by each algorithm:

Fuzzy learning rules (*algorithm proposed in* [3]) *test* 1:

R_0: IF X_3 is $\{\mathrm{HN}, \mathrm{MN}\}$ THEN Y is Iris Setosa

R_1: IF X_2 is NOT $\{\mathrm{MP}\}$ and X_3 is $\{Z, \mathrm{SN}\}$ THEN Y is Iris Versicolor

R_2: IF X_1 is NOT $\{\mathrm{HN}, \mathrm{MN}\}$ and X_2 is NOT $\{\mathrm{MP}, \mathrm{HP}\}$ and X_3 is NOT $\{\mathrm{MN}, \mathrm{HN}, \mathrm{MP}\}$ THEN Y is Iris Versicolor

R_3: IF X_0 is Z and X_1 is HN and X_2 is SP and X_3 is SP THEN Y is Iris Versicolor

R_4: IF X_1 is NOT $\{\mathrm{SN}, \mathrm{MN}\}$ and X_2 is NOT $\{\mathrm{MP}, \mathrm{HP}\}$ and X_3 is NOT $\{\mathrm{MN}, \mathrm{HN}, \mathrm{SP}\}$ THEN Y is Iris Versicolor

R_5: IF X_0 is NOT HN and X_1 is NOT HN and X_2 is NOT $\{\mathrm{MP}, \mathrm{HP}\}$ and X_3 is NOT $\{\mathrm{MP}, \mathrm{MN}, \mathrm{HN}\}$ THEN Y is Iris Versicolor

R_6: IF X_1 is NOT SN and X_2 is NOT SP and X_3 is $\{\mathrm{SP}, \mathrm{SN}\}$ THEN Y is Iris Versicolor

R_7: IF X_3 is HP THEN Y is Iris Virginica

R_8: IF X_2 is NOT SP and X_3 is $\{\mathrm{MP}, \mathrm{HP}\}$ THEN Y is Iris Virginica

Table 3
Reduction of fuzzy rules

Test	Without reduction	With reduction
1	70.3/129	66.2/15
2	66.2/127	66.2/21

R_9: IF X_0 is NOT Z and X_1 is NOT $\{SN, Z\}$ and X_3 is $\{SP, MP, HP\}$ THEN Y is Iris Virginica
R_{10}: IF X_0 is Z and X_1 is HN and X_2 is SP and X_3 is SP THEN Y is Iris Virginica
R_{11}: IF X_1 is NOT Z and X_3 is $\{MP, HP\}$ THEN Y is Iris Virginica
R_{12}: IF X_2 is NOT $\{SP, MP\}$ and X_3 is $\{SP, MP, HP\}$ THEN Y is Iris Virginica
R_{13}: IF X_1 is NOT MN and X_2 is NOT SP and X_3 is $\{SP, MP, HP\}$ THEN Y is Iris Virginica
R_{14}: IF X_2 is NOT $\{Z, SP\}$ and X_3 is $\{Z, MP, HP\}$ THEN Y is Iris Virginica

Reduction of fuzzy learning rules, *test* 1:
R_0: IF X_3 is $\{HN, MN\}$ THEN Y is Iris Setosa
R_1: IF X_3 is $\{SN, Z\}$ THEN Y is Iris Versicolor
R_2: IF X_2 is NOT HP and X_3 is $\{SN, Z, SP\}$ THEN Y is Iris Versicolor
R_3: IF X_3 is $\{MP, HP\}$ THEN Y is Iris Virginica
R_4: IF X_2 is NOT SP and X_3 is $\{SP, MP, HP\}$ THEN Y is Iris Virginica

The Pima Indians Diabetes Database has been used in [7]. In the test with this database, we use 576 instances (75%) for learning and 192 instances (25%) for proving the rules that have been learnt. Most of the attribute values in the examples from the training set appear very closely grouped together. This fact complicates the database.

After performing the proposed algorithm with the Pima Indians Diabetes Database, we obtained the results that are shown in Table 3.

The suggested algorithm drastically reduces the number of rules obtained. The rules obtained will be those that hold the evident knowledge from the set of examples that show general ideas of the expert reasoning.

In all tests, the division of the input variable domains is fixed. We use some set of labels established beforehand. This means that the results obtained will be worse than if the algorithm searches dynamically the optimal division of the input variables domains as is carried out by some machine learning techniques.

The suggested algorithm for acquiring the evident knowledge present in a set of expert's past actions has: $O(m)$ time complexity when ordering the input variables with respect to their correlation with the output variable; $O(m^2)$ time complexity to calculate P^- and remove the existing noise from the training set; $O(m \times n)$ time complexity to make some genes nonmutable; $O(m^2 \times n \times L)$ time complexity to obtain the set of initial rules (algorithm [3]); $O(NG \times N \times m)$ time complexity to apply the "genetic algorithm" and $O(N)$ time complexity in to remove the unnecessary rules, where m is the number of training examples, n is the number of input variables, L is the number of fuzzy labels, N is the number of rules generated and NG is the number of iterations of the genetic algorithm. Hence the time complexity of the algorithm is $O(NG \times N \times m)$, where $N < m$ and NG will be the lowest, if we only apply one genetic algorithm.

6. An example of how to use the rules obtained to extend the acquired knowledge

In this section, we attempt to give an approximation about how to use the rules obtained with the target of improving knowledge from the expert.

The set of rules generated provides a mechanism for interviewing domain experts. The rules will be used for facilitating the interview process. Thus, we save the knowledge engineer one of the greatest problems in knowledge acquisition that is learning enough about the domain to guide the interview process effectively.

One example of a strategy that we can apply is the *differentiation strategy*. This strategy is well known to knowledge engineers and has been applied by some knowledge acquisition tools as a method for acquiring more knowledge [12]. Differentiation implies seeking rules with different outputs, which come into conflict, that is situations in which these rules of different outputs may be applied at the same time and resolving those conflicts.

For example, in the set of fuzzy rules:

R_0: IF X_3 is $\{HN, MN\}$ THEN Y is Iris Setosa
R_1: IF X_3 is $\{SN, Z\}$ THEN Y is Iris Versicolor

R_2: IF X_2 is NOT HP and X_3 is $\{SN, Z, SP\}$ THEN Y is Iris Versicolor
R_3: IF X_3 is $\{MP, HP\}$ THEN Y is Iris Virginica
R_4: IF X_2 is NOT SP and X_3 is $\{SP, MP, HP\}$ THEN Y is Iris Virginica

We have a conflict with the rules R_2 and R_4. One example with the following form, X_0 and X_1 are some labels belonging to the set $\{HN, MN, SN, Z, SP, MP, HP\}$, X_2 is a label of the set $\{HN, MN, SN, Z, MP\}$ and X_3 is SP, can be classified by two rules of different classes.

At this point, we can ask the expert how he resolves this conflict, and we can suggest two possibilities:

- Introduction of a new variable that allows us to resolve this conflict.
- Introduction of a meta-rule, for example *if the rules R_i and R_j come into conflict R_i applies because it classifies the most common class*. Thus, we resolve the conflict.

The expert must choose the solution to the conflict. We only suggest a set of possible solutions.

7. Conclusions

In this paper, we have developed a general method for obtaining the evident knowledge existing in a set of training examples that shows the way the expert reasons. For doing so, the algorithm makes use of an inductive learning algorithm and a genetic algorithm that uses fuzzy logic.

The genetic algorithm has been conceived as a mechanism for increasing rule legibility obtained by means of other learning algorithms (we have applied it to [3], but it could be applied to other learning algorithms) so that those rules can be used in later phases of the knowledge acquisition process. The genetic algorithm by means of the generalization increases rule legibility.

This technique will acquire the more general knowledge which may be used for designing an interview with an expert who will extend, update and improve that set of obtained rules by answering the questions.

The main focus of our research is to show the knowledge acquisition process as a process for extending, updating and improving an incomplete knowledge base, in which machine learning is useful.

Some aspects of this algorithm will be studied in future work, such as, the study of strategies that guide the interview process making use of the rules obtained so that the tool acquires information that will lead to a stronger performance on the part of the system. The study of new forms of knowledge and methods of learning to deal with it will also be interesting. In the phase of defining a preliminary knowledge base, the expert is to be allowed to introduce into KB whatever knowledge he is able to express easily, the tool developed must also allow him to define other forms of knowledge.

References

[1] S. Abe, M. Lan, A method for fuzzy rules extraction directly from numerical data and its application to pattern classification, IEEE Trans. Fuzzy Systems 3 (1) (1995) 18–28.
[2] J.M. Benitez, A. Blanco, I. Requena, An empirical procedure to obtain fuzzy rules using neural network, Proc. 6th IFSA Congress, 1995.
[3] J.L. Castro, J.J. Castro-Schez, J.M. Zurita, Learning maximal structure rules in fuzzy logic for knowledge acquisition in expert systems, Fuzzy Sets and Systems 101 (1999) 331–342.
[4] J.L. Castro, M. Delgado, F. Herrera, A learning method of fuzzy reasoning by genetic algorithms, Proc. 11th European Congress of Fuzzy and Intelligent Technologies, 1993.
[5] J.L. Castro, J.M. Zurita, An inductive learning algorithm in fuzzy systems, Fuzzy Sets and Systems 89 (1996) 193–203.
[6] M. Delgado, A. Gonzalez, An inductive learning procedure to identify fuzzy systems, Fuzzy Sets and Systems 55 (1993) 121–132.
[7] R. Fisher, The use of multiple measurements in taxonomic problems, Annu. Eugenics 7 (2) (1936) 179–188.
[8] D. Goldberg, Genetic Algorithms in Search, Optimizations, and Machine Learning, Addison-Wesley, New York, 1989.
[9] A. Gonzalez, A learning methodology in uncertain and imprecise environments, Internat. J. Intell. Systems 10 (1995) 357–371.
[10] L. Hall, A. Kandel, Designing fuzzy expert systems, Verlag TÜV Rheinland, Germany, 1986.
[11] G. Hwang, Knowledge acquisition for fuzzy expert systems, Internat. J. Intell. Systems 10 (1995) 541–560.
[12] G. Kahn, S. Nowlan, J. McDermott, Strategies for Knowledge Acquisition, IEEE Trans. Pattern Anal. Mach. Intell. 5 (1985) 511–522.
[13] A. Kandel (Ed.), Fuzzy Expert Systems, CRC Press, Boca Raton, FL.
[14] R. Michalski, A theory and methodology of inductive reasoning, in: Machine Learning: An Artificial Intelligence Approach, vol. I, Morgan Kaufman, Los Altos, CA, 1983, pp. 83–134.

[15] J. Quilan, P. Compton, K. Horn, L. Lazarus, Inductive knowledge acquisition: A case study, Proc. 2nd Australian Conf. on Applications of Expert Systems, in: J.R. Quilan (Ed.), Applications of Expert Systems, Academic Press, Maidenhead, in press.

[16] S. Sestito, T. Dillon, Knowledge acquisition of conjunctive rules using multilayered neural network, Internat. J. Intell. Systems 8 (1993) 779–805.

[17] J. Smith, J. Everhart, W. Dickson, W. Knowler, R. Johannes, Using the ADAP learning algorithm to forecast the onset of diabetes mellitus, Proc. Symp. on Computer Applications and Medical Care, IEEE Computer Society Press, Silver Spring, MD, 1988, pp. 261–265.

[18] G. Tecuci, Automating knowledge acquisition as extending, updating, and improving a knowledge base, IEEE Trans. Systems Man Cybernet. 22 (6) (1992) 1444–1460.

[19] L. Wang, J. Mendel, Generating fuzzy rules by learning from examples, IEEE Trans. Systems Man Cybernet. 22 (6) (1992) 1414–1427.

[20] L. Zadeh, The role of fuzzy logic in the management of uncertainty in expert systems, Fuzzy Sets and Systems 11 (1983) 197–227.