Mathware & Soft Computing 3 (1996) 371-382

Neural Methods for Obtaining Fuzzy Rules*

J.M. Benítez, A. Blanco, M. Delgado & I. Requena
Dept. of Computer Science and Artificial Intelligence
E.T.S. Computer Engineering. Granada University
Avda. Andalucía, 38. 18071 - Granada
e-mail: {jmbs,requena}@robinson.ugr.es

Abstract

In previous papers, we presented an empirical methodology based on Neural Networks for obtaining fuzzy rules which allow a system to be described, using a set of examples with the corresponding inputs and outputs. Now that the previous results have been completed, we present another procedure for obtaining fuzzy rules, also based on Neural Networks with Backpropagation, with no need to establish beforehand the labels or values of the variables that govern the system.
Keywords: Obtaining fuzzy rules, Learning, Artificial Neural Networks.

1 Introduction.

A fundamental problem and one with multiple applications in Artificial Intelligence is that of identifying and reproducing the behaviour of systems. Amongst the main areas of its application are to be found: the acquisition of knowledge and that of determining causal relationships.

The most widely used method for representing knowledge is the one based on rules. In many real problems, describing the system by using fuzzy rules has advantages, either because the description is more appropriate for the real situation, or because the underlying information in the system is imprecise or fuzzy, using fuzzy techniques as an efficient tool for handling uncertainty.

In this paper some methods are described for carrying out the automatic acquisition of knowledge, using Artificial Neural Networks (ANN). On the one hand, a methodology presented previously in [BEN94, BEN95], and on the other hand, another methodology which uses, in the first phase, the ideas proposed in [SES93] for extracting classic production rules. Furthermore, a method of this kind may help in the study of ANN, by attempting to offer an interpretation of their internals.

The features of the ANN, such as their capacity for learning and generalizing using examples, and their robustness against noise, make them particularly efficient for gathering the information contained in a set of data and reproduce them

*This work has been partly supported by project PB 92-0945 from DGICYT. Madrid

faithfully. Therefore the knowledge attained by an ANN may be expressed through the learning of a set of samples in the form of fuzzy rules.

2 ANN for acquiring knowledge.

In [SES93], a method was proposed for obtaining a system's classic production rules, of the following kind:

$$R_i: \text{ If } a_1^i \wedge a_2^i \wedge \ldots \wedge a_{n_i}^i \text{ then } b_i$$

This rule states that if attributes a_j^i are presented in an example then a b_i class object is involved (they consider binary variables). The rules must be as simple as possible, i.e., they must be sufficient to characterize the object and have the smallest number of attributes in the antecedent. The key idea is that of training a multilayer ANN by incorporating its own outputs as extra inputs and studying the relationship between the weights of the real inputs and those of the outputs (extra inputs) with the hidden layer. Essentially the method may be summed up by:

1. An ANN with a single hidden layer is trained, in which the outputs are also used as extra inputs. That is to say, if the system's data have n inputs and m outputs, the training examples have $n + m$ inputs and m outputs. The Backpropagation algorithm is used.

2. For each pair (input i, output j), a measure of the agreement or proximity between the weights of these neurons to the hidden layeris calculated, the closer the relationship is the smaller it shall be:

$$\text{SDCP}_{ij} = \sum_{k=1}^{h} (w_{ik} - w_{jk})^2$$

3. A new set of samples is obtained from the examples, by changing the inputs for their complement to 1, that is, input i is changed for 1 - i. An ANN with no hidden layers is trained with this set of samples and by using Hebb's rule. Thus, inhibiting relationships are obtained between the inputs and outputs. The measure of the correlation between an input and an output shall be its own weight V_{ij} in the trained network.

4. The relationship between each input and output is measured by means of

$$prod_{ij} = \text{SDCP}_{ij} \cdot V_{ij}$$

5. For every output, the inputs are ordered according to the growth direction of the values of these products. A suitable cutoff point is established and the attributes which remain below the cutoff point are considered to be relevant. The rule is constructed by combining the attributes selected as relevant in the antecedent and putting the output in the consequent.

The method may easily be extended to the case in which attributes with a finite number of possible values are considered. In [SES93], some possibilities are pointed out for the case in which there is no suitable cutoff, which would indicate more than one rule for that particular output.

In [SES93], this method is applied to the example of the 7 segment LED, showing good results.

In [SES92], instructions are also given for extending the method to the case with continuous attributes. In this case the production rules take the form:

$$R_i: \text{ If } (v_{min}{}_1^i \leq a_1^i \leq v_{max}{}_1^i) \wedge \ldots \wedge (v_{min}{}_{n_i}^i \leq a_{n_i}^i \leq v_{max}{}_{n_i}^i) \text{ then } b_i$$

In this version of the method, all of the steps from the previous case are carried out, but in step 5 only the structure of each rule is obtained (which we shall call pre-rule), thereby indicating the presence of relevant attributes. Suitable ranges for these attributes are established by following the steps below:

1. Find the examples covered by each pre-rule. For this purpose a measure of the sum of the square differences is used and a degree ϵ_i is assigned to each rule. This degree may be different for different rules and, in fact, it is dynamically modified until it manages to ensure that all the examples are covered by a rule.

2. For every attribute included in the rule, its validity range is established as the interval for which the extremes are the minimum and maximum of that attribute's values for the examples covered by the rule.

3 Obtaining fuzzy rules.

In many real situations the system has to be described using fuzzy rules (e.g., the data is vague and is better represented using fuzzy numbers) or it provides gain because fuzzy rules represent the system more appropriately.

We set out two methodologies for ANN. One was first described in a previous paper, based on the prior construction of all the possible rules, then selecting the best combination thereof. The second one, which is based partly on the method described in Section 2, constructs rules using ANN trained with the examples.

3.1 Selection Methodology (descendant).

In [BEN94], we presented a methodology for obtaining fuzzy rules to describe a system from a set of examples of its behaviour and which basically consists of:

1. We suppose that we have identified the relevant variables in the description of the system's behaviour, the values of which are labels. The labels are represented by trapezoidal fuzzy numbers that are processed in the ANN as described in [REQ95].

   We train an ANN (EN) with a set of examples of the system's behaviour. The real value in the examples are processed as singleton fuzzy numbers.

2. We construct all the possible rules, according to the labels (values) of the variables identified.

3. We place each rule's antecedents into the network and we compare the latter's output with the consequent. We reject the rules that clearly do not adapt to the system (EN) and we obtain the first selection of relevant rules in the description thereof.

4. From amongst all the possible combinations of preselected rules, the combination which best adapts to the ANN (EN) and which has the lowest cardinal is taken as the definitive set of rules.

In order to reduce the computing time in step 4, in [BEN95] we used a greedy algorithm. The underlying idea is simple: it involves starting out from an empty set of rules and then adding on the best rule from the ones remaining in each step, until the set reaches the desired criterion.

Let the set of samples $M = \{e_j\}$, and $R = \{R_i\}$ the set of rules (step 3). The l-th input component of the j-th example is x_j^l and the k-th of output y_j^k. The rules are presented in the form:

$$\text{If } X_1 \text{ is } A_1 \text{ and } \ldots \text{ and } X_n \text{ is } A_n \text{ then } Y \text{ is } B$$

Definitions. **Degree of Coverage**[1] of the rule R_i over the sample e_j:

$$Gc(R_i, e_j) = 1 - \frac{\sqrt{\sum_{l=1}^{t}(IP(A_i^l) - x_j^l)^2 + \sum_{k=1}^{s}(IP(B_i^k) - y_j^k)^2}}{MaxD},$$

where IP(D) is the value of the Average Index[2] of fuzzy set D [CAM89, GON90]; and MaxD is the maximum value for the expression in the numerator. $Gc(Ri, ej)$ indicates the degree in which R_i represents e_j.

Partial Degree of Coverage of a set of rules C and a rule, R_i:

$$GPc(C, R_j) = \sum_{k=1}^{|M|} \begin{cases} Gc(R_j, e_k) - Gc(C, e_k), & \text{if } Gc(R_j, e_k) - Gc(C, e_k) > 0 \\ 0, & \text{otherwise} \end{cases}$$

GPc indicates by how much the coverage of the set of rules is improved when R_j is added to it. $GPc(C, R_j)$ is zero when a rule does not improve the coverage of the set.

In each step, the greedy algorithm selects the rule which makes the coverage of the set the greatest possible, or rather, the rule with the highest partial degree of

[1] Another definition of the degree of coverage adjusted to the system of inference or control being studied may be:

$$Gc(R_i, e_j) = 1 - \frac{d(y_j, S(R_i, x_j))}{MaxD}$$

where d is a metric in the output space and $S(R_i, x_j)$ represents the fuzzy system's output for x_j when its only rule is R_i.

[2] Another index function on fuzzy numbers could be used instead of the Average Index.

coverage. The process ends when all the samples (or a high percentage of them, β) are covered to a degree above a certain level, α. We may express the stop criterion as when the average degree of coverage is greater than or equal to β. With these two parameters we can regulate the precision which we demand of the description which returns the algorithm. The algorithm is now expressed as:

1. $C \leftarrow \{R_i\}$ such that $GMc(R_i)$ is maximum.
2. While $((GMc(C) < \alpha\beta)$ and $(|C| < |R|\ ))$
   $C \leftarrow C \cup \{R_j\}$ such that $R_j \notin C$ and $GPc(C, R_j)$ is maximum.
3. The set of rules is C.

The algorithm produces a result that is quasi-optimal and may be improved by modifying step 1 in order to construct several sets of rules by starting out from a different rule each time and, in the end, keep the best one from all those obtained.

Neither of the two versions ensures obtaining the optimal set of rules but they certainly are pretty fast. The algorithmic complexities are $O(|M| \cdot |R|^2)$ for the first version and $O(|M| \cdot |R|^3)$ for the second, which are quite small when compared to an exhaustive search, which would involve $O(|M| \cdot 2^{|R|})$.

3.1.1 Experimental Results.

In order to check the performance of the method for obtaining rules, we applied it to the problem of a car's braking system [WOL91]. This involves finding the force that has to be applied to a car to keep it at one point by taking into account its position and speed. Morover, random forces with varying modulus and sense have to be taken into account. The problem may be solved with a fuzzy controller which uses a base of seven rules with the following structure:

$$R_i\text{: If } p \text{ is } A_i \text{ and } v \text{ is } B_i \text{ then } f \text{ is } C_i$$

where p is the position, v is the speed and f is the force. These are three linguistic variables that take values in sets with a granularity of 5, 3 and 5, respectively.

In order to apply step 1 from the method, we use 250 control examples in which real values are represented as singleton fuzzy numbers. With these examples and the Backpropagation algorithm we train a network with 8 input neurons and 4 output neurons.

Applying step 2 produces 75 possible rules which, after carrying out step 3, are reduced to 15.

In order to obtain the definitive set of rules we employ the greedy algorithm, using different values for the degree of coverage. By varying the value of this degree between 1 and 0.77, we have obtained sets of rules of cardinals between 11 and 2. To illustrate this, look at the following examples: for a degree of 0.812 the set has 7 rules and for a degree of 0.8, the set is made up by 4.

We can establish the level of precision which we demand from the description by modifying the average degree of coverage accordingly, which may be translated into the degree of effectiveness in the case of fuzzy rules for control.

In accordance with the results obtained in the experiments, the following comments and conclusions may be made:

- The set of 7 rules obtained with an average degree of 0.812 is different from that obtained in [WOL91]. Nevertheless, the control of the system is equally flawless. With the sets of 5 and 4 rules, good control is still achieved, though the maximum external force that may be controlled is slightly reduced.

- The algorithm takes advantage of the neural networks' properties of generalization and robustness in order to obtain fuzzy rules from a set of examples. The main aim when drawing up the rules is to obtain a description of the system's behaviour, but they are also useful for fuzzy control. The precision of the description given may be regulated by means of two parameters.

- The method is independent from the way in which the linguistic labels are established. Bearing in mind that the main objective is obtaining a description of a process which is meaningful for a human expert, the set of labels must have a cardinal limited by the granularity detected and the form of its membership functions must meet some requirements like those described in [LEE90].

- A greedy algorithm is used in the final step which offers a fast and simple solution. Alternatively, some kind of combinatorial optimization technique might be used such as Simulated Annealing, the Boltzmann Machine or Genetic Algorithms.

- In real cases in which the involvement of human experts is important for the descriptions of coverage, a personal decision-making index associated with the expert would be used, as is described in [REQ95].

3.2 Constructive Methodology (ascendant).

A description of the knowledge included in a set of data which is closer and more intelligible for a human being is the rule of production. Systems based on fuzzy rules are, the same as ANN, universal approximators [CAS95] and, furthermore, they can justify the answers they give by supporting them on their rule base.

The case of data in which the attributes are continuous is extremely common, however representation using the rules that are generated with the method described in [SES92] is not appropriate. We believe that an approximation more in line with reality would be to use fuzzy rules, by considering each attribute as a variable, the possible values of which are fuzzy sets in the definition interval. On the other hand, once the rules have been established, the sets may be labeled thereby giving the rules a more appropriate aspect for their use and understanding by human beings. This aspect is of crucial importance if we attempt to create a system that would offer us justified answers.

With this new outlook, a construction method was designed which expresses the knowledge contained in a set of data in the form of fuzzy rules.

Firstly, we consider that the data obtained from the system and which we shall use in the training of the ANN, are expressed as real numbers (crisp), normalized in [0,1]: nevertheless, the variables, which take part in the rules which shall describe the system, are to be fuzzy and evaluated with linguistic labels expressed as trapezoidal fuzzy numbers.

The first approximation we suggest is the following:

1. The set of data is transformed by normalizing the inputs and outputs. An ANN is trained with a single hidden layer, adding the outputs as extra inputs.

2. For every input i and output j (in the network's input), we calculate

$$\mathrm{S}_{ij} = \sum_{k=1}^{h} (w_{ik} - w_{jk})^2$$

   where k goes through the neurons of the hidden layer.

3. The set of data is transformed by standardizing the inputs and outputs. The inputs are also changed by their complement to 1. An ANN, without hidden layers, is trained using Hebb's rule. The trained network's weights are now V_{ij}.

4. The relationship between every input and output is measured using:

$$prod_{ij} = S_{ij} \cdot V_{ij}$$

   For each output j, these products are placed in order from the lowest to the highest.

5. We look for an appropriate cutoff point in these ordered products and we consider the inputs i (variables) which remain below the cutoff point as contributing to the output j. Thus we obtain what we may consider as the structure of a rule or a pre-rule for each output j, expressed in terms of the variables as:

$$\mathrm{P}_{R_i} : X_p^i, X_q^i, X_r^i \rightarrow Y_j^i$$

6. In order to obtain the labels associated with the variables in the pre-rule we carry out the following:

   - In the training examples we only consider the component parts of the variables that are in the pre-rule. Using these subvectors we make a fuzzy clustering, based on the proximity of the values of Y_j (or which might also have close values in the X_i in the pre-rule). For this rule we reject the examples that it does not classify well.
   - With the examples from each class we construct the label's membership function for each variable. Thereby we obtain the same amount of effective rules and classes in the clustering.

A simple way is to consider triangular or trapezoidal numbers. For a variable, the support would be given by its minimum and maximum values in the examples of that kind. The mode would be given by the median (quartiles 1 and 3, for trapezoidal ones) of the values of that variable in the class examples). Another way is obtaining a linear approximation of the distribution of frequencies in those values of the variable).

Secondly, we consider that the system's examples are expressed by labels or fuzzy numbers (trapezoidal ones (a, b, c, d)[3] without any loss of generality). In this case the method may be applied as well, but with changes in the following steps:

- In step 1, we use the processing of the fuzzy numbers in the network as described in [REQ92, REQ94], but also considering the outputs as inputs.
- In step 2, in order to calculate S_{ij}, each addend is now obtained as

$$\frac{((\mathrm{a}(e_i)-\mathrm{a}(s_j))^2+(\mathrm{b}(e_i)-\mathrm{b}(s_j))^2+(\mathrm{c}(e_i)-\mathrm{c}(s_j))^2+(\mathrm{d}(e_i)-\mathrm{d}(s_j))^2)}{4}$$

  where $\mathrm{x}(e_i)$ represent the weights of the i-th input fuzzy number for the corresponding inputs (a, b, c, d) which represent the fuzzy number, and $\mathrm{x}(s_j)$ indicates the same thing for the j-th output fuzzy number placed as an input.
- In step 3, the V_{ij} are obtained in the same way as in the previous point.

The remaining steps in the process are carried out in the same way.

3.2.1 Experimental Results.

To illustrate the method, let us consider a simple example. This is the well-known iris plant problem. The goal is to recognize the type of the iris plant to which a given individual belongs. There are three classes of plants: setosa, versicolor and virginica. The data set is composed of 50 instances per class giving a total of 150. One class is linearly separable from the other two; while the latter are not linearly separable from each other. Each instance features four continuous attributes: petal length, petal width, sepal length, and sepal width. We coded the three possible classes as three values, so every sample was composed of seven components.

In the first stage, which is obtaining the relevant input for each output we use two neural networks. The first one had 7, 3 and 3 neurons in input, hidden and output layers, respectively. The second one only has input and output layers with 4 and 3 units in each one. After the training, both nets have reached a good enough learning level. Then we proceed to obtain the so-called prerules. The following ones were attained:

[3] We represent fuzzy trapezoidal numbers by four parameters (a, b, c, d) where $[a, d]$ is the support set and $[b, c]$ is the mode of the fuzzy set

1. *petal-length* → *setosa*

2. *sepal-width*, *petal-length*, *petal-width* → *versicolor*

3. *petal-length*, *petal-width* → *virginica*

Next, we proceed to establish the fuzzy sets for each variable. For each prerule, the subset of sample data which conforms to was selected, and then a clustering method was applied to. We considered the Chiu method [CHI94] method, which produces both the number of cluster and their centers. Then the examples are assigned to the cluster which center is the closest. Each cluster gives rise to a rule which has the structure (antecedents and output) given by the prerule and the antecedents fuzzy values as fuzzy trapezoidal numbers calculated out of the cluster members (cluster with 1 or 2 examples are not considered). In order to obtain the rules, the data are split in two parts, in a random way (100 examples to obtain the rules and 50 to test the rules). This process was applied several times. Because of the random process, each time the rules can be slightly different. So we had the following results, expressed in percentage of hits:

Experiment	Number of rules	Training Set	Test set
The worst	4	97.0	88.0
The average	5	96.0	94.0
The best	4	96.0	96.0

In the best case, we have obtained the following rules:

1. If *petal-lenght* is (1.06, 1.30, 1.60, 1.94) → *setosa*

2. If *sepal-width* is (1.93, 2.50, 3.00, 3.47) and *petal-length* is (2.89, 4.00, 4.60, 5.20) and *petal-width* is (0.96, 1.20, 1.50, 1.84) → *versicolor*

3. If *petal-length* is (4.76, 5.00, 5.60, 5.64) and *petal-width* is (1.35, 1.80, 2.20, 2.45) → *virginica*

4. If *petal-length* is (5.65, 5.80, 6.30, 6.75) and *petal-width* is (1.76, 1.90, 2.30, 2.54) → *virginica*

Also, we have used the 150 examples to obtain the rules. Then we have obtained five rules and 96% of hits over all the examples.

So the obtained results are good in general. We have checked that *setosa* plants are always classified correctly, and mistakes being committed with the other two classes.

We think that better results could be obtained with a different way of performing the clustering.

4 Conclusions and final comments.

Two methodologies have been described for obtaining fuzzy rules from a system, about which a set of examples of its behaviour is known, on the basis of possibly relevant variables or attributes in its behaviour.

Both methodologies consider, in the first place, the training of an ANN, using the backpropagation algorithm and the examples available from the system. Both methodologies may be used, both whether the examples of the system are given as crisp values or as fuzzy numbers (e.g., labels).

The first methodology ([BEN95]) starts out from the construction of all the possible rules and selects the relevant ones in the description of the system; that is why we point it out as a selective or descendant methodology. We have presented the experimental results obtained from a fuzzy control example and they are quite satisfactory.

The second methodology extends a method for obtaining rules of production with binary attributes [SES93] to obtain fuzzy rules which may allow the system to be described. It determines the input variables which contribute to every possible output and constructs fuzzy rules; that is why we describe it as a constructive methodology.

Even though initial results look promising, the obtained rules may be tuned to improve their performance. That is why we are considering alternative ways of obtaining the corresponding labels for variables (step 6) as:

a) With the data from the system (examples) a fuzzy clustering is made, in accordance with the experts' criteria and/or those of the system to be solved. For each variable in the system's output, by using the coverage degree defined in [BEN95] (see Section 3.1), we obtain the examples covered by the pre-rule in each class (label) of the output variable, Y, above a (preset) level α, and labels are obtained the same way before.

b) Other way is to obtain, for each example, the minimun level α in which the rule cover to each example, for each label of the variable Y. Then, the same process as above is carried out, except that the rules obtained are dependent on level. In the latter case, we are considering two possibilities:

   - With a degree of precision α, each pre-rule covers a given set of examples. We construct the membership function with these examples for each variable (its label) in the rule. This is done by considering the median (triangular numbers) or 2 percentiles (trapezoidal ones). This gives a good overall set of rules at the level of α. We believe this is better than the previous one.
   - With a degree of precision α, each example shall or shall not be covered by a rule. The example's degree of membership (i.e., from the value of the corresponding attribute) is the lowest of these α. Construct the membership function using these membership degrees.

c) To use a fuzzy clustering method to obtain the rules.

Finally, we are also considering the distribution of information by using several networks trained with backpropagation. Each network would learn the examples of a pre-rule, which have been previously clustered.

References

[BEN94] Benítez, J.M., Blanco, A., and Requena, I. *Un procedimiemto empírico con Redes Neuronales para obtener reglas difusas.* IV Congreso de la Asociación Española de Tecnologías y Lógica Fuzzy. Blanes, Septiembre, 1994.

[BEN95] Benítez, J.M., Blanco, A., and Requena, I. *An Empirical Procedure to Obtain Fuzzy Rules using Neural Networks.* 6th IFSA Congress. Brasil, July, 1995.

[CAS95] Castro, J.L. *Fuzzy Logic Controllers are Universal Approximators.* IEEE Trans. on Syst. Man and Cybern. **25**(4), April. (1995).

[CAM89] Campos, L.M. and González, A. *A Subjective Approach for Ranking Fuzzy Numbers.* Fuzzy Sets and Systems, **29**, 145–153. (1989).

[CHI94] Chiu, S.L.*A cluster estimation method with extension to Fuzzy Model Identification.* IEEE Intern. Congress on Fuzzy Systems. pp.1240–1245. (1994).

[GON90] González, A. *A study of the ranking function approach through mean values.* Fuzzy Sets and Systems, **35**, 29–41. (1990).

[LEE90] Lee, C.C. *Fuzzy Logic in Control Systems: Fuzzy Logic Controller.* IEEE Trans. on Systems, Man, and Cybernetics. **20**(2), Part I, 404–419, Part II, 419–435. (1990).

[REQ92] Requena, I. Ph. D. Thesis: *Redes Neuronales en Problemas de Decisión con Ambiente Difuso.* Depto. Ciencias de la Computación e I.A. University of Granada. (*In Spanish*). (1992).

[REQ94] Requena, I., Delgado, M., and Verdegay, J.L. *Automatic Ranking of Fuzzy Numbers with the Criteria of a Decision-maker Learnt by an Artificial Neural Network.* Fuzzy Sets and Systems, **64**, 1–19. (1994).

[REQ95] Requena, I., Blanco, A., Delgado, M., and Verdegay, J.L. *A Decision Personal Index to Fuzzy Numbers based on Artificial Neural Networks.* Fuzzy Sets and Systems. (To appear)

[SES92] Sestito S. and Dillon, T. *Automated Knowledge Acquisition of Rules with Continuously Valued Attributes.* Proc. Avignon'92. (1992).

[SES93] Sestito S. and Dillon, T. *Knowledge Acquisition of Conjunctive Rules Using Multilayered Neural Networks.* Inter. J. Intelligent Systems, 8, 779–805. (1993).

[WOL91] Wolf, T. *Das Fuzzy-Mobil.* mc 3/91, pp. 50–63. (1991).