Automatic Tuning of Fuzzy Partitions in Inductive Reasoning

Angela Nebot

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Jordi Girona Salgado, 1-3,
Barcelona 08034, Spain
angela@lsi.upc.es

Abstract. The aim of this research is to automatically tuning a good fuzzy partition, i.e. determine the number of classes of each system variable, in the context of the Fuzzy Inductive Reasoning (FIR) methodology. FIR is an inductive methodology for modelling and simulate those systems from which no previous structural knowledge is available. The first step of FIR methodology is the fuzzification process that converts quantitative variables into fuzzy qualitative variables. In this process it is necessary to define the number of classes into which each variable is going to be discretized. In this paper an algorithm based on simulated annealing is developed to suggest a good partition in an automatic way. The proposed algorithm is applied to an environmental system.

1 Introduction

The Fuzzy Inductive Reasoning (FIR) methodology emerged from the General Systems Problem Solving (GSPS) developed by Klir [1]. FIR is a data driven methodology based on systems behavior rather than structural knowledge. It is a very useful tool for modelling and simulate those systems from which no previous structural knowledge is available. FIR is composed of four main processes, namely: *fuzzification*, *qualitative model identification*, *fuzzy forecasting*, and *defuzzification*. Figure 1 describes the processes of FIR methodology.

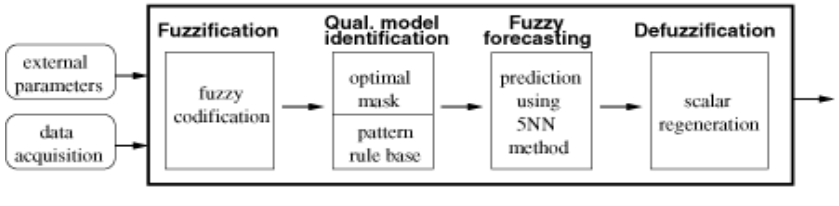

Fig. 1. FIR structure

The fuzzification process converts quantitative data stemming from the system into fuzzy data, i.e. qualitative triples. The first element of the triple is the

A. Sanfeliu and J. Ruiz-Shulcloper (Eds.): CIARP 2003, LNCS 2905, pp. 554–562, 2003.

class value, the second element is the fuzzy membership value, and the third element is the side value. The side value indicates whether the quantitative value is to the left or to the right of the peak value of the associated membership function.

The model identification process is able to obtain good qualitative relations between the variables that compose the system, building a pattern rule base that guides the fuzzy forecasting process.

The fuzzy forecasting process predicts systems behavior. The FIR inference engine is a specialization of the k-nearest neighbor rule, commonly used in the pattern recognition field.

Defuzzification is the inverse process of fuzzification. It makes possible to convert the qualitative predicted output into a quantitative variable that can then be used as input to an external quantitative model. It has been shown in previous works that FIR methodology is a powerful tool for the identification and prediction of real systems, specially when poor or non structural knowledge is available [2,5]. For a deeper insight into FIR methodology the reader is referred to [4].

As can be seen in figure 1, for the fuzzification process of FIR methodology to start it is necessary to define some external parameters, i.e. the partition (number of classes of each system variable) and the landmarks (limits between classes). The default value for the number of classes' parameter for each system variable is three and the equal frequency partition (EFP) is used as the default method to obtain the landmarks of the classes. These default values have been used in different applications obtaining usually good results. However, experience has shown that in some them, i.e. biomedical and ecological, the determination of the partition parameter needed in the fuzzification step becomes relevant for the identification of a good model that captures systems behavior in an accurate way. The automatic determination of a good partition as a pre-process of FIR methodology is an interesting and useful alternative. To achieve this goal an algorithm base on simulated annealing is presented in this paper and used in an environmental application, i.e. prediction of ozone concentration in a specific area of Mexico city. The algorithm proposed is introduced in section 2. In section 3 the ozone application is addressed and the results obtained discussed. Finally, the conclusions of this research are given.

2 Simulated Annealing Algorithm

Simulated annealing is a generalization of a Monte Carlo method that was introduced by Metropolis et al. in 1953 [3]. This technique is used to approximate the solution of very large combinatorial optimization problems and is based on the manner in which liquids freeze in the process of annealing [7]. In an annealing process a melt, initially at high temperature and disordered, is slowly cooled so that the system at any temperature is approximately in thermodynamic equilibrium. Cooling proceeds until the final temperature is reached, that corresponds to the most stable (lowest energy) system state. If the initial temperature of the

system is too low or cooling is not done sufficiently slowly the system may be trapped in a local minimum energy state.

A simulated annealing algorithm consists of two nested loops. The outer-most loop sets the temperature and the inner-most loop runs a Metropolis Monte Carlo simulation at that temperature. The algorithm starts with an initial solution to the problem, which is also the best solution so far and a value for an initial high temperature. Each iteration consists of the random selection of a new solution (called candidate solution from now on) from the neighborhood of the current one. The cost function of the candidate solution is evaluated and the difference with respect to the cost function value of the current solution is computed (δ in equation 1). If this difference is negative, i.e. the cost function value of the candidate solution is lower than the one of the current solution, the candidate solution is accepted. If the difference is positive the candidate solution is accepted with a probability based on the Boltzmann distribution (equation 1).

$$f_{Boltzmann}(\delta) = \exp(-\delta/k.T) \tag{1}$$

where T is the temperature value and k is the Boltzmann's constant. The accepted candidate solution becomes the current solution and if its cost function value is lower than the one of the best solution, this one is updated. If the candidate solution is rejected, i.e. the Boltzmann probability is less than the random number generated, the current solution stays the same and it is used in the next iteration. The temperature is lowered in each iteration down to a *freezing* temperature where no further changes occur. The set of parameters that determine the temperature decrement is called the cooling schedule. This parameters are the initial temperature, the function that decrements the temperature between successive stages, the number of transitions needed to reach the quasi-equilibrium for each temperature value and the stop criterion.

The main aspects to be considered in a simulated annealing implementation are: 1) *solution configuration*, 2) *new solutions generation mechanism*, 3) *cost function* and 4) *cooling schedule*. These aspects, for the algorithm proposed in this paper, are explained in detail next while the algorithm is shown in figure 2.

Solution Configuration

The solution should contain the number of classes for each variable. The configuration chosen is a vector with the same number of columns than the number of system variables, containing integers in the range $[2 \cdots maxNC]$. $maxNC$ the maximum number of classes allowed.

New Solutions Generation Mechanism

Two options can be used to generate the initial partition, i.e. current solution. The first one sets all the variables to 3 classes (default option in the current FIR implementation). The second one corresponds to a random generation of the number of classes for each system variable.

The procedure to generate a new solution, i.e. candidate solution, from the current one is to increment or decrement by one the number of classes associated to a certain system variable. The variable that is going to be modified is chosen randomly from the vector of variables. The decision to increase or decrease the

number of classes of this variable is also random. If the extremes are reached, i.e. 2 or $maxNC$, it is enforced to apply the increment and decrement operators, respectively.

Cost Function

An important aspect in this research is to define an appropriate cost function for the evaluation of the partitions. To address this issue it is necessary to look closer to the qualitative model identification processe of FIR methodology.

In the process of modeling, it is desired to discover the causal and temporal relations between the inputs and the output of the system, that make the resulting state transition matrix as deterministic as possible. The more deterministic the state transition matrix is, the higher is the likelihood that the future system behavior will be predicted correctly. In FIR, the causal and temporal relations among the fuzzy qualitative variables are represented by a *mask* matrix. Equation 2 gives an example of a mask,

$$\begin{array}{c} {}_t\backslash^x \\ t-2\delta t \\ t-\delta t \\ t \end{array} \begin{array}{c} \begin{array}{cccc} i_1 & i_2 & i_3 & O \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & -2 & -3 & 0 \\ -4 & 0 & 0 & +1 \end{pmatrix} \end{array} \tag{2}$$

where δt indicates the sampling period. A mask denotes a dynamic relationship among qualitative variables. The negative elements represents the causal relations between he inputs and the output (positive value in the mask). The sequence in which they are enumerated is immaterial. In position notation the mask of equation 2 can be written as $(4, 6, 7, 9, 12)$, enumerating the mask from top to bottom and from left to write.

A *quality* value, based on an entropy reduction measure, is computed for each mask considered. In [4] the quality function is described in detail. The mask with highest quality is called the optimal mask. It is important to note that the optimality of the mask is evaluated with respect to the identification (training) data set. Therefore, the best mask is not, necessarily, the one that achieves the best forecast of the test data.

In this study the quality function that evaluates the information associated to the mask is used as the cost function. In that way, no prediction is needed in the partition evaluation process. Therefore, only the fuzzification and the model identification processes of FIR methodology (see figure 1) are executed to compute the cost function for a specific partition. This reduces considerably the execution time of the simulated annealing algorithm proposed.

Cooling Schedule

Let us now take a look to all the parameters that conform the cooling schedule. The *initial temperature* depends on the initial solution generated and it is computed using equation 3,

$$T_0 = \frac{\mu}{-\ln(\Phi)} \cdot Cost(S_0) \tag{3}$$

where $Cost(S_0)$ evaluates the cost function of the initial solution (S_0) and $\mu, \Phi \in [0, 1]$ and their values depend on the number of variables of the application as described in equation 4.

$$\mu = 0.3, \Phi = 0,3 \; if \; N \leq 3 \qquad \mu = 0.1, \Phi = 0,1 \; if \; N > 3 \tag{4}$$

Equation 4 says that initially it is possible to accept solutions μ per one worse than the initial solution with a probability Φ.

Two different *cooling functions* are predominantly used, i.e. linear and proportional. In this work, the proportional cooling function proposed by Kirpatrick [7] is used to decrement the temperature between successive stages. This function is presented in equation 5.

$$T_{k+1} = \alpha \cdot T_k \qquad with \;\; \alpha = 0.9 \tag{5}$$

The *number of transitions* needed to reach the quasi-equilibrium for each temperature is defined by means of two values, the maximum number of transitions i.e. iterations in the inner loop and the maximum number of accepted solutions. The maximum number of iterations is set to N^3 and the maximum number of accepted solutions is set to N^2, being N the number of system variables.

Three *stop criterions* have been used in this study. The simulation annealing algorithm stops when the number of iterations is grater than the maximum number of possible solutions ($masxNC^N$), the last iteration has finished with no accepted solutions and/or N iterations have been completed without an enhancement of the global solution, i.e. the best solution is not changed during the last N iterations. It is important to remark here that if the algorithm stops due to the first criterion no advantage is obtained with respect to an exhaustive search. Moreover, the annealing algorithm do not guarantee that the optimal solution is found. The main algorithm is presented next.

```
function [BestSol] = Annealing (N,maxNC,data)

 % A first solution (Current Solution) and an initial temperature (T) are
 % generated. The evaluation of the cost function for the initial partition
 % is also computed and stored in the CurrentSol structure
 [CurrentSol,T] = GenerateCurrentSol(N,maxNC,data);


 % The Current Solution is the Best Solution so far
 BestSol = CurrentSol;

 % The Current Solution is stored in the list of generated solutions
 SolList = [CurrentSol];

 % The total number of solutions generated is set to one
 NumberSolutions = 1;

 % Initialization of both the number of iterations without a global
 % enhancement and the number of iterations without accepted solutions.
 IterNoGlobalEnhance = 0;
 IterNoAcceptedSol = 0; % boolean variable

 % Loop that sets the temperature
 while (NumberSolutions <= (maxNC^N)) & (IterNoGlobalEnhance < N)
        & (~IterNoAcceptedSol),
```

```
  % Initialization of the boolean variable that establishes if a global
  % enhancement has been produced, the number of accepted solutions and
  % the number of iterations for the current temperature
  GlobalEnhance = 0; % boolean variable
  NumAcceptSol = 0;
  NumIter = 0;


  % Loop that runs the Metropolis Monte Carlo simulation
  while (NumIter < N^3) & (NumAcceptSol < N^2),


    % The number of iterations is incremented
    NumIter = NumIter + 1;


    % The GenerateCandidateSol function generates the Candidate Solution.
    % If this solution is not in the list of generated solutions, evaluates
    % its cost function and includes both values in the solution list. If the
    % solution is already in the list (it has been generated one or more times
    % in the past), the cost function is available and it is not computed again.
    [CandidateSol,SolList] = GenerateCandidateSol(N,maxNC,CurrentSol,SolList,data);


    % The total number of solutions generated is incremented
    NumberSolutions = NumberSolutions + 1;


    % The difference between the cost function of the Candidate Solution and
    % the cost function of the Current Solution is stored in the Delta variable
    Delta = CandidateSol.cost - CurrentSol.cost;


    % Condition for the acceptance of the Candidate Solution
    if (rand(1) < exp(- Delta/T)) | (Delta < 0)


      % When accepted, the Candidate Solution becomes the Current Solution
      CurrentSol = CandidateSol;
      % The number of accepted solutions is incremented
      NumAcceptSol = NumAcceptSol + 1;
      % If the Current Solution has a lower cost function value than
      % the one of the Best Solution, this one is actualized
      if (CurrentSol.cost < BestSol.cost)
        BestSol = CurrentSol;
        GlobalEnhance = 1; % boolean variable
      end;
    end;
  end;
  % The temperature is decremented
  T = alpha*T;


  % The IterNoAcceptedSol and IterNoGlobalEnhance variables are actualized
  % once the quasi-equilibrium is reached for the current temperature
  IterNoAcceptedSol = (NumSolAcept == 0);
  if GlobalEnhance
     IterNoGlobalEnhance = 0;
  else IterNoGlobalEnhance = IterNoGlobalEnhance + 1;
  end;
end;
return
```

Fig. 2. Simulated Annealing algorithm for the automatic determination of fuzzy partitions in FIR methodology (Implemented in Matlab 6.5 languaje)

3 Ozone Concentration

The main air pollution problem that has been identified in Mexico city metropolitan area (MCMA) is the formation of photochemical smog, primarily ozone (O_3). High levels of ozone causes eye irritation, respiratory disorders, crop damage and increased deterioration rate of material. In these circumstances, it is important and useful to provide early warnings of high levels of ozone concentration so that the authorities can react as fast as possible. Therefore, the construction of ozone models that capture the behavior of this gas in the atmosphere as precisely as possible is of interest not only for environmental scientists but also for government agencies. There are many different models available for local scale predictions of air quality and for ozone level forecasting. In recent years paradigms such as neural networks [8], decision trees or association rules [9] have been used for this purpose. In [6], FIR methodology has been used to model the ozone contaminant in the centre region of the Mexico city. Seven variables are involved in this study. The input variables are hour of day hd (from 0 to 23), day of week dw (from 1 to 7), wind speed ws, measured in meters per second (m/s), wind direction wd, measured in degrees (from 0° to 359°), temperature t, measured in $°C$ and relative humidity hu, measured in percentage (from 0% to 100%). The ozone $o3$ (measured in parts per million (PPM)), is the system's output variable. Ozone and weather data were available from January to May 2000 and contain missing values. The data of the first four months is used as identification data set, whereas the month of May is used as test data set. The mean square error in percentage (MSE) is used to determine the validity of each of the models.

In [6] three different partitions have been studied to find the model with the best prediction performance. The best optimal mask found, performing an exhaustive search, for the three partitions studied are presented in the first three rows of the table 1.

Table 1. Partitions results obtained by the previous work (first three rows) and the Simulated Annealing algorithm (las three rows)

Partition *hd dw ws wd t hu o3*	Optimal Mask	Quality	MSE test
(3, 3, 3, 3, 3, 3, 3)	(1,14,21)	0.595	52.08%
(5, 5, 4, 5, 4, 4, 4)	(4,14,17,21)	0.537	145.7%
(6, 6, 2, 3, 3, 4, 2)	(1,14,21)	0.738	39.36%
(3, 2, 6, 4, 6, 5, 2)	(1,14,17,21)	0.736	34.90%
(3, 3, 4, 2, 4, 2, 2)	(1,14,17,21)	0.763	38.33%
(3, 2, 5, 2, 3, 2, 2)	(1,14,17,21)	0.757	38.45%

In table 1, the first column contains the number of classes for each variable (partition). The second column describes the optimal mask associated to

that partition, in position notation. The third column contains the quality of the optimal mask, i.e. the cost function in the simulated annealing algorithm proposed. The last column contains de MSE prediction error of the test data set. The last 3 rows shows the partitions proposed by the simulated annealing algorithm when running it several times (more than 20). As can be observed from table 1 the partitions chosen by "hand" in the previous work have lower quality and performance than the partitions suggested by the simulated annealing algorithm. The partitions recommended by the SA algorithm have similar qualities and performances, and any of them can be used as a good partition parameter in the fuzzification process of FIR methodology. To chose a partition without previous criterion is a big risk that the modeler can avoid by using the SA algorithm presented. Clearly, the SA algorithm is a very useful tool that allows the modeler start using FIR in a more efficient way.

4 Conclusions

In this paper, a simulated annealing algorithm for the automatic tuning of fuzzy partitions in the context of the fuzzy inductive reasoning methodology has been presented. The SA algorithm suggests, for each variable, the number of classes to be discretized, basing its decision on the quality of the best mask associated to that partition. The use of the SA algorithm for the modeling of the ozone contaminant in Mexico city shows the potentiality of this approach.

Acknowledgements. The author thanks the suggestions of F.E. Cellier, F. Mugica and P. Villar during the development of this research. This research was supported by Spanish Consejo Interministerial de Ciencia y Tecnología (CICYT), under project DPI2002-03225.

References

1. G. Klir. *Architecture of Systems Problem Solving*. Plenum Press, New York, 1985.
2. F. Mugica and F. Cellier. Automated synthesis of a fuzzy controller for cargo ship steering by means of qualitative simulation. In *Proc. ESM'94, European Simulation MultiConference*, pages 523–528, Barcelona, Spain, 1994.
3. M. R. N. Metropolis, A. Rosenbluth and E. T. A. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21:1087–1092, 1953.
4. A. Nebot. *Qualitative Modeling and Simulation of Biomedical Systems Using Fuzzy Inductive Reasoning*. Ph.d. thesis, Dept. Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya, 1994.
5. A. Nebot, F. Cellier, and D. Linkens. Synthesis of an anaesthetic agent administration system using fuzzy inductive reasoning. *Artificial Intelligence in Medicine*, 8(3):147–166, 1996.
6. A. Nebot, F. Mugica, and P. Gómez. Long term prediction of maximum ozone concentration using fuzzy inductive reasoning. In *Proceedings EUNITE'01: European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems*, pages 91–101, Tenerife, Spain, 13–15 december 2001.

7. C. D. G. J. S. Kirkpatrick and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.
8. D. Wieland and W. Wotawa. Local maximum ozone concentration prediction using neural networks. In *Proceedings of the AAAI Workshop On Environmental Decision Support Systems and Artificial Intelligence*, pages 47–54, 1999.
9. F. Wotawa and G. Wotawa. From neural networks to qualitative knowledge in ozone forecasting. *AI Communications*, 14:23–33, 1993.