Annales Univ. Sci. Budapest., Sect. Comp. 12 (1991) 151-158

COMPUTATIONAL COMPLEXITY OF VARIOUS FUZZY INFERENCE ALGORITHMS

László T. Kóczy
Dept. of Communication Electronics,
Technical University of Budapest,
Budapest, Sztoczek 2, H-1111, Hungary

Abstract: Essence of most industrial applications of fuzzy reasoning and control is some kind of rule based fuzzy inference algorithm. One of the crucial points in these algorithms is the computational speed i.e. time complexity (usually in the uniform complexity sense). This paper shows the complexity of three different algorithms.

Keywords: Fuzzy rule based inference, compact rule algorithm, computational complexity

1. INTRODUCTION

A large number of industrial applications of fuzzy logic and fuzzy systems have appeared in recent years on the market. Reason of this is the fact that even modern control theory has failed to cope with some classes of control problems in industrial processes, robotics, vehicles, household equipment, video cameras, etc. etc. The problem is that classical control theory describes and models only a limited class of not very complex systems properly.

The basic idea of fuzzy algorithms and especially rule based fuzzy inference was initially proposed by Zadeh (see Zadeh (1968, 1974, 1975) and many other papers). The first results of practical implementation are connected with the name of Mamdani (see Mamdani (1974, 1977), King & Mamdani (1977) etc.). At present time dozens of various concrete algorithms based on the original idea however using more or less different technics are represented in the industrial fuzzy systems. Leading are here some Japanese schools (cf. Sugeno & Nishida (1985), Hirota & al. (1987), etc. etc.).

2. APPROXIMATE REASONING BY FUZZY RULES

What is the essence of approximate reasoning modelled by fuzzy inference? Let us take the example of driving a car, a task that can be managed by most grown-up people quite succesfully, although none of us uses any exact algorithm in his head while sitting at the volant. And yet the fully automatic control of driving a car in a real traffic environment has still not yet been solved.

Modelling the 'algorithm' used by the driver, an approach is the description by a set of rules accumulated during the period of learning and later, during 'sharp' driving. Examples for such rules are '*Drive* **slower** *if the street is wet*' or '*Push the brake* **firmly** *if you are* **near** *to a red light*'. Such instructions are natural language statements containing linguistic terms like **firmly**, **near**, etc. Linguistic terms can be represented by fuzzy sets over the observation (condition) and the conclusion space ('If' and 'Then' part, resp.). The result of combining observation with premise rules is a set of observation dependent fuzzy rules i.e. fuzzy relations, which are further combined with each other and so describe a fuzzy set in the conclusion space. This result can be used then either as a fuzzy conclusion or by some method (e.g. center of gravity) a crisp conclusion is calculated.

3. TWO BASIC ALGORITHMS

In the following, we give two fundamental algorithms one of which is applied in the majority of the industrial systems.

The form of every rule in the system is

$$R_1 = 'If\ X\ is\ C_i\ then\ Y\ is\ S_j'.$$

In the above C_i and S_j are fuzzy sets over X and Y, respectively, both of them representing a fuzzy term or a combination of terms. In a general case both X and Y are constructed as the direct product of k_1 and k_2 component spaces, respectively, i.e.

$$X = X_1 \times X_2 \times \ldots \times X_{k_1} \text{ and } Y = Y_1 \times Y_2 \times \ldots \times Y_{k_2}$$

Terms C_i and S_j are composed similarly of k_1 and k_2 simple terms over $X_1, \ldots, X_{k_1}$ and $Y_1, \ldots, Y_{k_2}$, resp.

The rule system is a set of rules like R_1 and in addition we have an observation A that is taken over X, and so is, similarly to C_i, a linguistic term in the observation space, i.e. a fuzzy set of X. Our primary purpose is to obtain a fuzzy cunclusion B over the conclusion space Y. Formally:

$$(R_1, R_2, \ldots, R_r; A) \Leftrightarrow B$$

The two algorithms are the following:

Algorithm 1

For $i = 1$ to r

let $m_i = \max_x\{\min\{C_i(x), A(x)\}\}$

let $S_i^A(y) = \min_y\{m_i, S_i(y)\}$

Let $B(y) = 0$

For $i = 1$ to r

let $B(y) = \max\{B(y), S_i^A(y)\}$

For $i = 1$ to k_2

$$\text{let } b_i = \frac{\sum_{Y-Y_i} y_i B(x,y)}{\sum_{Y_i} y_i}$$

B is the fuzzy set obtained as the union of all truncated conclusions of the rules. The crisp conclusion is b the center of gravity of the area below $B(y)$.

Algorithm 2

For every (x, y)

let $R(x,y) = 0$

For $i = 1$ to r

let $R_i(x,y) = \min\{C_i(x), S_i(y)\}$

let $R(x,y) = \max\{R(x,y), R_i(x,y)\}$

For every (x, y)

let $B(x,y) = \min\{A(x), R(x,y)\}$

For $i = 1$ to k_2

$$\text{Let } b_i = \frac{\sum_X \sum_{Y-Y_i} y_i B(x,y)}{\sum_{Y-i} y_i}$$

Here, R_i are the compact rules which are represented as fuzzy relations in $X \times Y$. R is the compact rule system including all information in $\{R_1, R_2, \ldots, R_r\}$. $B(x,y)$ is the fuzzy conclusion this time in the form of a fuzzy relation and b is the crisp conclusion as 'spacial' center of gravity.

4. THE PROPERTIES OF THE ALGORITHMS

To compare practical usability of the above two methods we have done some examinations by comparing the behaviour of both algorithms in respect of identical changes in the premises, the results in Kóczy & Hirota (1991) show that Algorithm 2 is 'better' than the other.

We cannot forget, however, computational speed, which influences applicability very seriously. For measuring the speed we use the uniform complexity model described e.g. in Aho et al. (1974).

We have r rules in the form

'If X is C_i then Y is S_j', so that

$X = X_1 \times \ldots \times X_{k_1}$ and $Y = Y_1 \times \ldots \times Y_{k_2}$,

$\#X_i = m_i \leq m$, $\#Y_j = n_j \leq n$; so

$\#X = \Pi m_i \leq m^{k_1}$, $\#Y = \Pi n_j \leq n^{k_2}$.

We calculate now the input complexity. There are r rules, each consisting of an 'If-' and a 'Then-part'. A fuzzy membership function in ak_i dimensional finite cardinality ($= m_j$) space is represented by $k_i m_j$ membership degrees (in the jth dimension m_j degrees). As m_j is limited from above by m and n, resp., the resulting complexity of the whole rule system is

$$I_1 = r(k_1 m + k_2 n)$$

Similarly, the complexity for a single observation in X is

$$O_1 = k_1 m$$

For the crisp conclusion first we obtain the minimum of the observation with the condition parts of the rules. For a single rule m_j steps are needed in the jth dimension. Then the maximum of all degrees over X_j takes an identical number of steps. This results into $2k_1 m$ in total. 'Truncation' in Y means n_j operations and unioning the result conditions the same (for every Y_j). In addition we have the steps necessary to calculate the center of gravity. Here, the elements in Y_j must be multiplied by the corresponding membership degree (of the conclusion part) and added (for the weighted sum), further on, the elements of Y_j must be added for the divider. The last step is the division, resulting the jth component of the center of gravity. If now m_j is estimated by m and n_j by n, the resulting complexity is

$$C_1 = r(2k_1 m + 2k_2 n) + k_2.3n + k_2$$

It is known from the literature that the execution of algorithms like 2 is exponential, so we give only the exact result. (For comparison, $I_1 = I_2$.)

$$C_2 = (r+1)m^{k_1} n^{k_2} + k_2(2m^{k_1} n^{k_2 - 1} + n + k_2)n + k_2)$$

C_1 and C_2 can be compared easier if we introduce $k = \max\{k_1, k_2\}$ and $N = \max\{m, n\}$. So

$$I_1 = I_2 = 2\text{rkN} = O(rkN)$$

$$C_1 = 2\text{rkN} + 3kN + k = O(rkN)$$

$$C_2 = (r+1)N^{2k} + k(2N^{2k-1} + N + k) = O(rN^{2k})$$

The result clearly shows that Algorithm 2 is very slow.

5. A FAST ALGORITHM WITH COMPACT RULES

Exponentiality can be eliminated in Algorithm 2 (compact rule method) if the following restrictions are accepted:

$$\#\sup(proj_{Z_k}(\mu(\varsigma))) \leq LIN(L = o(\#\min\{X_i, Y_j\})$$

where $\varsigma = x$ or $y, \mu = C, S$ or A, $proj_{Z_k}$ stands for the projection of a membership function to $Z = X_i$ or Y_j, i.e. $\mu(x_i)$ or $\mu(y_j)$, and supp is the support. Also k_1 and k_2 must be kept constant, i.e. the number of linguistic variables in the rules is limited as well. So we have

Algorithm 3

Let $R(nil) = 0$

For $j = 1$ to k_1 let $d_j = u_j =$ nil

For $j = 1$ to k_2 let $D_j = U_j =$ nil

For i $= 1$ to r

For $j = 1$ to k_1

Mark $m_{i,j} = \min\{supp(proj_{X_j}(C_i))\}$

Mark $M_{i,j} = \max\{supp(proj_{X_j}(C_i))\}$

For $j = k_1 + 1$ to $k_1 + k_2$

Mark $m_{i,j} = \min\{supp(proj_{X_j}(S_i))\}$

Mark $M_{i,j} = \max\{supp(proj_{X_j}(S_i))\}$

For $(x, y) \in [m_i, M_i]$

Let $R(x, y) = R(x, y) \vee \min\{C_i(x), S_i(y)\}$

Comment* $[m_i, M_i]$ stands for

$[m_{i,1}, M_{i,1}] \times \ldots \times [m_{i,k_1}, M_{i,k_1}] \times [m_{i,k_1+1}, M_{i,k_1+1}] \times \ldots$

$\ldots \times [m_{i,k_1+k_2}, M_{i,k_1+k_2}]$,

so there are $k_1 k_2$ limited cycles nested into each other

$/ \min\{C_i(x), S_i(y)\} = R_i(x, y)$*

For $j = 1$ to k_1

if $d_j =$ nil then let $d_j = m_{i,j}$ else if $m_{i,j} < d_j < M_{i,j}$

then let $d_j = m_{i,j}$ else if $d_j < m_{i,j} < u_j$ then

nop else if $D_j < m_{i,j}$ then let $d_j = (d_j \rightarrow m_{i,j})$

if $u_j =$ nil then let $u_j = M_{i,j}$ else if $M_{i,j} > u_j > m_{i,j}$

then let $u_j = M_{i,j}$ else if $u_j > M_{i,j} > d_j$ then

nop else if $d_j > M_{i,j}$ then let $u_j = (u_j \rightarrow M_{i,j})$

Comment* $<$ and $>$ are understood as referring to any element in the chains d_j and u_j

respectively *

For $j = k_1 + 1$ to $k_1 + k_2$

if D_j= nil then let $D_j = m_{i,j}$ else if $m_{i,j} < D_j < M_{i,j}$

then let $D_j = m_{i,j}$ else if $D_j < m_{i,j} < U_j$ then

nop else if $D_j < m_{i,j}$ then let $D_j = (D_j \rightarrow m_{i,j})$

if U_j= nil then let $U_j = M_{i,j}$ else if $M_{i,j} > U_j > m_{i,j}$

then let $U_j = M_{i,j}$ else if $U_j > M_{i,j} > D_j$ then

nop else if $D_j > M_{i,j}$ then let $U_j = (U_j \rightarrow M_{i,j})$

Comment* < and > are understood as referring to any element in the chains D_j and U_j,

respectively *

For $(x, y) \in$

$[\min\{supp(proj_{X_j}(A))\}, \max\{supp(proj_{X_j}(A))\}]\ x[D_j, U_j]$

let $B(x, y) = R(x, y) \wedge A(x)$

For i = 1 to k_2

$$\text{let } b_i = \frac{\sum_X \sum_{Y-Y_i} y_i B(x, y)}{\sum_{Y_i} y_i}$$

The complicatedness of this algorithm is determined first of all by the necessity of administering the total support area in such a way that never the exponential size of the whole space is 'detected' in any step (including the start with an 'empty' space - in fact empty only in our records). A serious disadvantage of Algorithm 3 is that although it allows fast and 'good' reasoning, the boundedness of the fuzzy terms results in a universe of discourse thinly covered by rules. How to overcome this difficulty is the topic of some further investigations in the direction of rule interpolation which has been done in Kóczy and Hirota (1991).

In the above the computational complexity connected with a single rule is kept constant if the number of fuzzy variables is constant.Let us calculate now the complexity in this latter case.

Every rule is stored only over its support. As the support is limited in every dimension by the small constant L, size of the membership functions representing the rules is limited by $L^{k_1+k_2}$. For r rules this sizes is $rL^{k_1+k_2}$. The observation is limited by L in every component of X, however it is not limited in Y. In case of r rules in the system the maximal extent of the union (max, t-conorm) of the rules is rL in every Y_i(it is less if there is overlapping between the rules). So intersection (min) of rule system and observation takes $L^{k_1}(rL)^{k_2}$ individual steps (min or t-norm operations). Calculating the center of gravity in all of the k_2 components of Y happens by adding all membership degrees of B in all the dimensions but the one in question ($k_1 + k_2 - 1$ dimensions), k_1 of which are limited in size by L and

the rest by rL. These degrees however must be multiplied by the corresponding value in the chosen component of Y which products will be added. Similarly, the values in Y are added for obtaining the divider. Finally the division is executed and so one component of b is obtained. The resulting complexity is so

$$C_3 = rL^{k_1+k_2} + L^{k_1}(rL)^{k_2} + k_2(L^{k_1}(rL)^{k_2-1} + 2n + 1)$$

It is crucially important now that L and k_i are constants (and L is a small constant in respect of N). So L^{k_1} and L^{k_2} are constants, as well and the only nonlinear members in the above complexity expression are formed by the $(rL)^{exp}$ members. Maximum exponent is k_2, so C_3 is reduced to

$$C_3 = O(r^{k_2})$$

i.e. it is polynomial. This astonishing fact is the main statement of this paper.

It must be mentioned that there are some other ways to reduce the complexity of the basic algorithm represented by Algorithm 2 (compact rule method). In some of them the number of possible linguistic terms i.e. the possible shapes of membership functions over X_i and Y_j are limited. In such a case the calculations can be restricted to a binary search in an exponential size rule combination tree, where the height is however linear and so the time complexity of the algorithm is linear (or at least polynomial). Such research is done by Turksen (1990) and group.

It must be seen clearly that all the solutions where the original exponential complexity is reduced restrict in some way the generality of the algorithm. In case of Algorithm 1 an essential part of the information is lost (that is why the sensitivity is not so good) in case of Algorithm 3 it is the arbitrarity of the membership functions that is given up, although still infinite variations are allowed. In Turksen's method the number of membership function shapes is reduced to a finite number complexity is however even better.

REFERENCES

[1] Aho A.V., J.E. Hopcroft & J.D. Ullman, *The design and analysis of computer algorithms.* (Addison-Wesley, Reading etc. 1974.)

[2] Hirota K., Y. Arai & Sh. Hachisu, Real time fuzzy pattern recognition and fuzzy controlled robot-arm. *Prepr.2nd IFSA Congr. (Tokyo),* 1987, 274-277.

[3] King P.J., E.H. Mamdani, The application of fuzzy control systems to industrial processes. *Automatica***13**,(1977) 235-242.

[4] Kóczy L.T., K. Hirota, Fuzzy inference by compact rules. *Proc. of Int. Conf. Fuzzy Logic & Neur. Netw. IIZUKA '90,* (Iizuka, 1990), 307-310.

[5] Kóczy L.T., K.Hirota, Rule interpolation in approximate reasoning based fuzzy control. *Proc of 4th IFSA'91 Brussels, Engineering,*(Brussels, 1991) 89-92.

[6] Mamdani E.H. Application of fuzzy algorithms for the control of a dynamic plant. *Proc. IEE* **121**,(1974) No. 12, 1585-1588.

[7] Mamdani E.H. Application of fuzzy logic to approximate reasoning using linguistic synthesis. *IEEE Tr. Comp. C-26*, **12**(1977), 1182-1191.

[8] Sugeno M., M. Nishida, Fuzzy control of model car. *Fuzzy Sets and Systems* **16**(1985), 103-113.

[9] Turksen B.I., personal communication in Iizuka, Japan (1990).

[10] Zadeh L.A., Fuzzy algorithms. *Info. Contr.* **12**(1968), 94-102.

[11] Zadeh L.A., Fuzzy logic and approximate reasoning. *Memo. ERL–M479*. (Electronics Res. Lab., Univ. of Cal., Berkeley, 1974.)

[12] Zadeh L.A., The concept of a linguistic variable and its application to approximate reasoning. *Info. Sci.***8**(1975), 199-249.