PERIODICA POLYTECHNICA SER. EL. ENG. VOL. 42, NO. 1, PP. 155-172 (1998)

# RADIAL BASIS FUNCTION ARTIFICIAL NEURAL NETWORKS AND FUZZY LOGIC

N. C. STEELE* and J. GODJEVAC**

* Control Theory and Application Centre
Coventry University
email: NSTEELE@coventry.ac.uk
** EPFL Microcomputing Laboratory
IN-F Ecublens, CH-1015 Lausanne, Switzerland

Received: May 8, 1997; Revised: June 11, 1998

## Abstract

This paper examines the underlying relationship between radial basis function artificial neural networks and a type of fuzzy controller. The major advantage of this relationship is that the methodology developed for training such networks can be used to develop 'intelligent' fuzzy controlers and an application in the field of robotics is outlined. An approach to rule extraction is also described.

Much of Zadeh's original work on fuzzy logic made use of the MAX/MIN form of the compositional rule of inference. A trainable/adaptive network which is capable of learning to perform this type of inference is also developed.

*Keywords:* neural networks, radial basis function networks, fuzzy logic, rule extraction.

## 1. Introduction

In this paper we examine the Radial Basis Function (RBF) artificial neural network and its application in the approximate reasoning process. The paper opens with a brief description of this type of network and its origins, and then goes on to show one way in which it can be used to perform approximate reasoning. We then consider the relation between a modified form of RBF network and a fuzzy controller, and conclude that they can be identical. We also consider the problem of rule extraction and we discuss ideas which were developed for obstacle avoidance by mobile robots. The final part of the paper will consider a novel type of network, the Artificial Neural Inference (ANI) network, which is related to the RBF network and can perform max/min compositional inference. Making use of some new results in analysis, it is also possible to propose an adaptive form of this network, and it is on this network that current work is focused.

## 2. The RBF Network

The radial basis function network has its origins in approximation theory, and such an approach to multi-variable approximation has been developed as a neural network paradigm notably by BROOMHEAD and LOWE [2]. Surprisingly perhaps, techniques of multi-variable approximation have only recently been studied in any great depth. A good account of the state of the art at the start of this decade is given by POWELL in [1]. In the application to the approximation of functions of a single variable, it is assumed that values of the function to be approximated are known at a number, $n$, of distinct sample points, and the approximation task is to construct a continuous function through these points. If $n$ centres are chosen to coincide with the known data points then an approximation can be constructed in the form

$$f(x) = \sum_{i}^{n} w_i \phi_i(r_i) \, ,$$

where $r_i = \|x - c_i\|$ is the radial distance of $x$ from the centre $c_i$, $i = 1 \ldots n$, and $w_i$, $i = 1 \ldots n$ are constants. Each of the $n$ centres is associated with one of the $n$ radial basis functions $\phi_i$, $i = 1 \ldots n$. The method extends easily to higher dimensions when $r$ is usually taken as the Euclidean distance. With the centres chosen to coincide with the sample or data points, and with one centre corresponding to each such point, then the approximation generated will be a function which reproduces the function values at the sample points exactly. In the absence of 'noise' this is desirable, but in situations when this is present, steps have to be taken to prevent the noise being modelled at the expense of the underlying data. In fact, in most neural network applications, it is usual to work with a lesser number of centres than of data points in order to produce a network with good 'generalisation' as opposed to noise modelling properties, and the problem of how to choose the centres has then to be addressed. Some users take the view that this is part of the network training process, but to take this view from the outset means that possible advantages may be lost. The 'default' method for 'prior' centre selection is the use of the $K$-means algorithm, although other methods are available. In a number of applications, notably those where the data occur in clusters, this use of a set of fixed centres is adequate for the task in hand. In other situations, this approach is either not successful, or is not appropriate, and the network has to be given the ability to adapt these locations, either as part of the training process, or in a later tuning operation. The exact nature of the radial basis functions $\phi_i$, $i = 1 \ldots n$ is not usually of major importance, and good results have been achieved in a number of applications using Gaussian basis functions when

$$\phi_i(r_i) = e^{-\left(\frac{r_i}{\sigma_i}\right)^2} \, ,$$

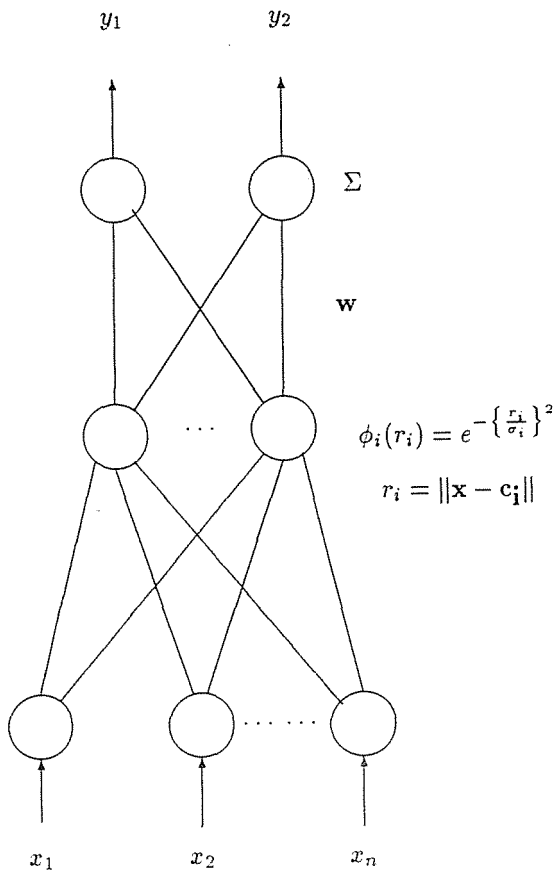


*Fig. 1.* The Radial Basis Function network

where $\sigma_i$, $i = 1 \ldots n$ are constants. This type of set of basis functions, with the property that $\phi \to 0$ as $r \to \infty$ is known as a set of localised basis functions, and although this appears to be a natural choice, there is no requirement for this to be the case. There are a number of examples of non-localised basis functions in use, including the thin-plate spline function,

$$\phi(r) = r^2 \ln(r)$$

and the multi-quadric function,

$$\phi(r) = (r^2 + \sigma^2)^{1/2}, \quad \sigma \text{ a constant.}$$

A major advantage of the RBF neural network, when the centres have been pre-selected, is the speed at which it can be trained. In *Fig. 1*, we show a network with two output nodes and we note that with the centres fixed,

the only parameters which have to be determined are the weights on the connections between the single hidden layer and the output layer. This means that the parameters can be found by the solution of a system of linear equations, and this can be made both efficient and robust by use of the Singular Value Decomposition (SVD) algorithm. The algorithm has further advantages when the number of centres is less than the number of data points since a least squares approximation is then found automatically.

If centres are not pre-selected, then some procedure for selection, perhaps based on gradient descent, must be incorporated into the training algorithm. In this case, either all parameters are determined by gradient descent, or a hybrid algorithm may be used, with the parameters of the centres being updated by this means and new sets of weights found (not necessarily at each iteration) again as the solution of a system of linear equations.

## 3. A Fuzzy Controller

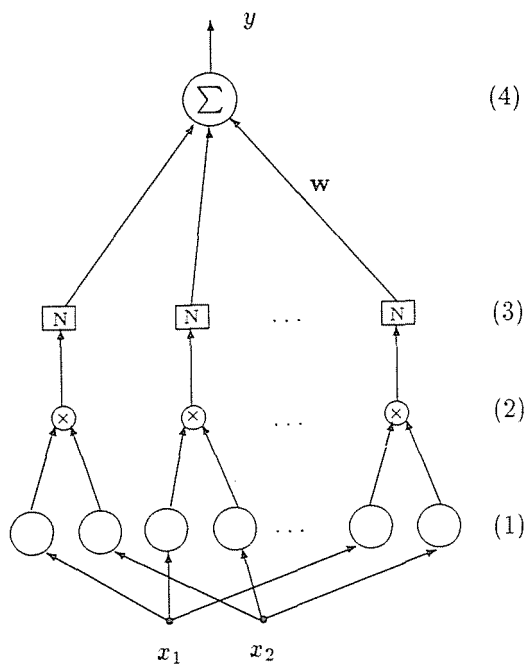


*Fig. 2.* A fuzzy controller

In *Fig. 2* we show the architecture of a fuzzy controller based on the Takagi–Sugeno design, with a crisp output. This controller is a variant of that discussed by JANG and SUN [4]. In general, this fuzzy controller has $m$ crisp

inputs $x_1, x_2, \ldots, x_m$ and, in this case, one output $y$. There are $n$ linguistic rules of the form:

$R_i$: IF $x_1$ is $A_{i,1}$ AND $x_2$ is $A_{i,2}$ AND ... AND $x_m$ is $A_{i,m}$ THEN $y$ is $y_i$ $i = 1, \ldots, n$ where $i$ is the index of the rule, $A_{i,j}$ is a fuzzy set for $i$-th rule and $j$-th linguistic variable defined over the universe of discourse for $j$-th variable, and $w_i$ is a real number. The fuzzy sets $A_{i,j}$, $i = 1, \ldots, n$, $j = 1, \ldots, m$ are each defined by Gaussian membership functions $\mu_{i,j}(x_j)$, where

$$\mu_{i,j}(x_j) = e^{-(x_j - c_{i,j})^2/\sigma_{i,j}} . \tag{1}$$

Here $c_{i,j}$ and $\sigma_{i,j}$ are the constant Gaussian parameters.

In the first layer of the structure shown in *Fig. 2*, fuzzification of the crisp inputs $x_1$ and $x_2$ takes place. This means that the degree of membership of the inputs $x_1$ and $x_2$ is calculated for each of the corresponding fuzzy sets $A_{i,j}$. These values are then supplied to be nodes in the second layer, where a $t$-norm is applied, and here we use the algebraic product. The output of the $k^{\text{th}}$ unit in this layer is then the firing strength $u_k$ of rule $k$, and in this case, with two inputs, it is given by

$$u_k = \mu_{k,1}(x_1)\mu_{k,2}(x_2) , \qquad k = 1, \ldots, n .$$

In the general case this is

$$u_k = \prod_{j=1}^{m} \mu_{k,j}(x_j) , \qquad k = 1, \ldots, n . \tag{2}$$

The rule firing strength may also be taken as a measure of the degree of compliance of the current input state with a rule in the rule base, taking the value 1 when compliance is total.

The overall network output at the fourth layer has to be a crisp value, and this is generated by the height method. The $w_i$, $i = 1, \ldots, n$ are the weights between the third and fourth layers and are in some sense 'partial consequents'. The output $y$ is finally formed as

$$y = \frac{\sum_{i=1}^{n} u_i w_i}{\sum_{i=1}^{n} u_i} . \tag{3}$$

This can be written as

$$y = \frac{u_1}{\sum_{i=1}^{n} u_i} w_1 + \frac{u_2}{\sum_{i=1}^{n} u_i} w_2 \ldots + \frac{u_n}{\sum_{i=1}^{n} u_i} w_n = \sum_{i=1}^{n} \overline{u}_i w_i . \tag{4}$$

*Eq.* (4) suggests that layer three should perform a normalisation function, and if this is the case the output of unit $l$ in layer three is given by

$$\overline{u}_l = \frac{u_l}{\sum_{i=1}^{n} u_i} .$$

The structure described here is reminiscent of a form of the RBF artificial neural network, and in earlier work GODJEVAC [5] gave an algorithm for training such a system. Using input/output training data, a set of weights $w_i$, $i = 1, \ldots, n$ are obtained, and values found for Gaussian parameters $c_{i,j}$ and $\sigma_{i,j}$ $i = 1, \ldots, n$, $j = 1, \ldots, m$, using a gradient descent algorithm. In effect, this means that the rule base is deduced from the input/output data. This work was carried out in connection with the robot obstacle avoidance problem, and good results were achieved for this application.

In [3] it was shown that the 'standard' RBF network could perform the function of an inference mechanism in fuzzy-logic based control. We now re-examine the RBF network in this rôle, with the aim of showing that the structure described above can be embedded in a generalisation of that paradigm. In order to do this, we consider the operation of the RBF network in detail. Following the presentation of the vector $\mathbf{x}$ as the input the expression for the output, $u_k$, of the $k$-th hidden node is,

$$u_k = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_k\|}{\sigma_k}\right)^2 \quad k = 1, \ldots, n$$

and can be written as

$$u_k = \prod_{j=1}^{m} \exp\left\{-\left(\frac{x_j - c_{k,j}}{\sigma_k}\right)^2\right\} = \prod_{j=1}^{m} \tilde{\mu}_{k,j}(x_j)\,, \tag{5}$$

say. There is some similarity between *Eqs.* (5) and (2). The similarity is not complete, however, since in the earlier case, the Gaussian variances were functions of two indices, and here they are functions of a single index. *Eq.* (5) shows that the output of the $k$th hidden unit in an RBF network can be viewed as the product of the degrees of membership of each component of $\mathbf{x}$ in fuzzy sets, with Gaussian membership functions *each with the same variance*, one centred on each co-ordinate of the centre vector $\mathbf{c}_k$. Again, this output value may be interpreted as a measure of rule firing strength.

In the next Section, we consider how the standard RBF model can be modified so as to reproduce the complete structure of the fuzzy system.

## 4. Modification of the Standard RBF Network

The network shown in *Fig. 3* is a modification of the standard RBF network in which a form of normalisation similar to that included in the fuzzy system described earlier. It should be noted that the addition of the new node attached to the hidden units by connections with unit weights, and the associated normalisation process, does not destroy the linearity of the optimisation task. The only change is to the right hand side of the systems of equations which have to be solved to determine the network weights.

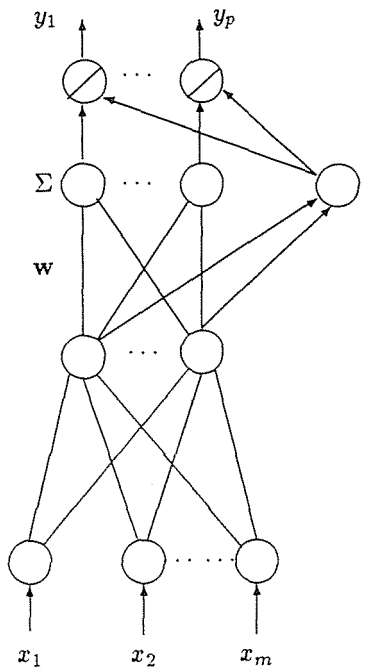


*Fig. 3.* RBF network with output normalisation

Network outputs would have to be modified as discussed in [3] to yield true fuzzy set membership functions, if this is what is required. However, as an alternative, the network could be required to perform the defuzzification operation itself, and such a network is shown in *Fig. 4*. In [3], vectors representing fuzzy sets were used as inputs. If instead, we use crisp state vectors, and if the state vectors corresponding to reference states as used in the rule base are used as centres, then the transition to the fuzzy reasoning system is virtually complete. The one remaining difference is, recalling (5), that the variance of the Gaussian activation function associated with each centre is a parameter whose value depends only on the centre or node number.

In order to represent the fuzzy reasoning system fully, this parameter must be allowed in addition to vary with each component of the input. To see how to do this within the modified RBF structure, consider again (2). If we set

$$\sigma_{k,j} = \sigma_{k,k}\lambda_{k,j} = \overline{\sigma}_k\lambda_{k,j} ,$$

where $\lambda_{k,j}$ is a parameter, this can be written

$$u_k = \exp\left\{-\sum_{j=1}^{m}\frac{(x_j - c_{k,j})^2}{\overline{\sigma}_k^2\lambda_{k,j}^2}\right\}$$

$$= \exp\left\{-\sum_{j=1}^{m}\frac{(\overline{x}_{k,j}-\overline{c}_{k,j})^2}{\overline{\sigma}_k^2}\right\}.$$

Here $\overline{x}_{k,j} = x_j/\lambda_{k,j}$, and $\overline{c}_{k,j} = c_{i,j}/\lambda_{k,j}$. This is now in the same form as (5), but note that we are using a coordinate scaling of the inputs and the centres, and that this scaling depends both on the associated centre and on the component of the input vector. The implication of this scaling operation is that we must allow weights on the arcs from the input nodes to the hidden units, with the centres also appropriately modified.

We have now achieved our aim of embedding the fuzzy reasoning system in the modified RBF architecture, however, there is a penalty. The new first-layer weights, or scaling factors, are additional adaptable parameters which also may have to be learnt during training. Because they occur within the arguments of the non-linear activation functions, the overall optimisation task is then no longer linear, and other methods must be used. This is examined further in the next Section.

## 5. Training and Adaptation

Given a set of rules for the rule base, and if we set the scaling factors to fixed values, then provided that we have sufficient input/output data, the modified network can be trained by solving a system of linear equations. From this basis, an algorithm can be deduced for adapting the network parameters to improve performance, based on the gradient descent method. This adaptation process would focus on modifying both the network weights, including the scaling parameters, and the Gaussian parameters. Changing the Gaussian parameters and scaling factors is equivalent to changing the rules in the rule base, since the fuzzy sets which describe them are modified by this process. This approach consists of distinct training and adaptation phases and given that the rules given will probably be imprecise, it is attractive to consider an approach which is based solely on the adaptive phase.

When training such a system for use as an obstacle avoidance controller for a mobile robot, a simple supervised learning approach, based on gradient descent was used to determine appropriate values of the parameters. With a known desired system output $y_d$ corresponding to an input vector $\mathbf{x}$ it is possible to define an error measure $E = (y_d - y)^2$, where $y$ is the actual response of the network. The partial derivatives $\partial E/\partial\alpha$ of $E$ with respect to the network parameters are calculated and parameters are adapted/updated according to the standard scheme

$$\alpha_{\text{new}} = \alpha_{\text{old}} - \theta_\alpha \frac{\partial E}{\partial \alpha}$$

with $\theta_\alpha$ a parameter dependent learning rate. Initially, all parameters were set to random values and using several input/output pairs, all parameters were adapted by this method. Other schemes are possible which exploit the partial linearity of the problem.

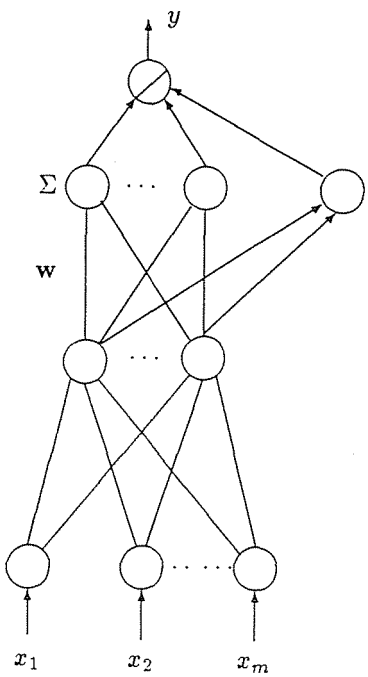


*Fig. 4.* Defuzzification included in the network: $y$ is now the defuzzified output

The methods outlined above were demonstrated successfully on trial problems involving function approximation. Experimental evidence showed that the fuzzy reasoning system, that is, the generalised RBF network, was capable of learning the data in fewer presentations. For the purpose of obstacle avoidance, both designs controlled the robot satisfactorily, although again the generalised network showed slightly superior learning speeds.

## 6. An Approach to Rule Extraction

The work described here also serves to suggest an approach to rule extraction from adaptive RBF networks, namely by interpreting them as fuzzy reasoning systems, and examining the final form of the centre vectors together with the associated values of $y$, the system output. It is tempting to try to place interpretations on the weights $w_i$, $i = 1, \ldots, n$, since these are playing the rôles of the locations of the maxima of the fuzzy sets which are

being combined if the height method for de-fuzzification is used. However, even if a single rule is being fired with strength 1, other rules will also usually be active, and these will contribute to the output $y$. This means that no single $w_i$ and thus a single fuzzy set, can be interpreted as an 'independent' consequent of rule $i$. We will thus describe an alternative approach for a network with a single output, but which generalises to multiple outputs in an obvious way.

For compactness, we use the notation $\phi_i(\mathbf{x}) = \phi_i(\|\mathbf{x}-\mathbf{c}_i\|)$, where $\|\mathbf{x}-\mathbf{c}_i\|$ is either the usual Euclidean distance, or a weighted form. Effectively we are working with normalised basis functions

$$\overline{\phi}_i(\mathbf{x}) = \frac{\phi_i(\mathbf{x})}{\sum_{j=1}^{N}\phi_j(\mathbf{x})} \qquad i = 1, \ldots, N\,,$$

where $N$ is the number of basis functions/centres. For the network output, we have

$$y = \sum_{i=1}^{N} w_i \overline{\phi}_i(\mathbf{x})\,.$$

Assume that the network has been trained and on completion the process, the set of centres $c_i$, $i = 1, \ldots, N$, and weights $w_i$, $i = 1, \ldots, N$ have been determined. We now establish $N$ network rules, $y_{R_i}$, $i = 1, \ldots, N$ by presenting the $N$ vectors corresponding to the $N$ centres as inputs to obtain as (crisp) outputs

$$y_{R_i} = w_1\overline{\phi}_1(\mathbf{c}_i) + \ldots + w_i\frac{1}{\sum_{j=1}^{N}\phi_j(\mathbf{c}_i)} + \ldots + w_n\overline{\phi}_N(\mathbf{c}_i)\,, \quad i = 1, \ldots, N\,.$$

That is,

$$\mathbf{y}_R = \begin{bmatrix} y_{R_1} \\ \vdots \\ y_{R_N} \end{bmatrix} = \overline{\mathbf{\Phi}}\mathbf{w}\,, \tag{6}$$

where

$$\overline{\mathbf{\Phi}} = \begin{bmatrix} \frac{1}{\sum_{j=i}^{N}\phi_j(\mathbf{c}_1)} & \cdots & \cdots & \cdots & \frac{\phi_N(\mathbf{c}_1)}{\sum_{j=i}^{N}\phi_j(\mathbf{c}_1)} \\ \frac{\phi_1(\mathbf{c}_2)}{\sum_{j=i}^{N}\phi_j(\mathbf{c}_2)} & \frac{1}{\sum_{j=i}^{N}\phi_j(\mathbf{c}_2)} & \frac{\phi_3(\mathbf{c}_2)}{\sum_{j=i}^{N}\phi_j(\mathbf{c}_2)} & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\phi_1(\mathbf{c}_N)}{\sum_{j=i}^{N}\phi_j(\mathbf{c}_N)} & \frac{\phi_2(\mathbf{c}_N)}{\sum_{j=i}^{N}\phi_j(\mathbf{c}_N)} & \frac{\phi_3(\mathbf{c}_N)}{\sum_{j=i}^{N}\phi_j(\mathbf{c}_N)} & \cdots & \frac{1}{\sum_{j=i}^{N}\phi_j(\mathbf{c}_N)} \end{bmatrix}$$

and $\mathbf{w}$ is the vector of weights.

When the network is operating and presented with an arbitrary input vector $\mathbf{x}$ the output is

$$\begin{aligned} y &= w_1\overline{\phi}_1(\mathbf{x}) + \ldots + w_N\overline{\phi}_N(\mathbf{x}) \\ &= [\overline{\phi}_1(\mathbf{x}), \ldots, \ldots, \overline{\phi}_N(\mathbf{x})]\mathbf{w}\,. \end{aligned}$$

So using *Eq.* (6) and assuming that $\overline{\Phi}$ is invertable,

$$y = [\overline{\phi}_1(\mathbf{x}), \ldots, \ldots, \overline{\phi}_N(\mathbf{x})]\overline{\Phi}^{-1}\mathbf{y}_R\,. \tag{7}$$

The entire operation of the fuzzy reasoning system RBF network is contained in this equation, from the process of fuzzification, through the calculation of the $t$-norm, inference and defuzzification to produce the system output. GODJEVAC [6] develop a method for the linguistic expression of rules obtained in this way, based on assigning primary labels, 'small', 'medium', and so on, to fuzzy sets on the antecedent universes of discourse. A set of hedges to operate on the membership functions was also defined, together with a measure of similarity between fuzzy sets. When network training is complete, the fuzzy set as defined by the Gaussian function associated with each component of each centre vector is examined. This is then given the label which corresponds to the closest of the (hedged) reference sets and by this means, all rule antecedent clauses can be established. There are two possible approaches to the consequent parts. In the approach developed by GODJEVAC, a further set of linguistic labels were assigned to elements of the consequent universe of discourse. This means that fuzzy rules with fuzzy sets as consequents can be deduced. However, since the underlying fuzzy system has *crisp* outputs, it may be more appropriate to extract rules of the form given in Section 3, with the consequent part stated as 'about $y_i$'. *Eq.* (7) provides further scope for investigation.

## 7. The Artificial Neural Inference Network (ANI-net)

This network carries out the reasoning process using the compositional rule of inference, and is shown in *Fig. 5*. In the first layer following the input nodes, the crisp system state vector is fuzzified using Gaussian membership functions as described earlier. If we wish to make the network adaptable, then any membership function which is at least piecewise differentiable with respect to its parameters, may be used instead. The nodes in this layer are grouped in such a way that each group calculates the degree of membership of the current state in one and only one (compound) rule antecedent clause. Thus if there are $N$ rules, and the dimension of the state vector is $n$, there will be a total of $nN$ nodes in this layer.

In the next layer, a $t$-norm operation is carried out by the nodes, and this is a MIN operation. Elsewhere, where network adaptation is required,

the $t$-norm has been taken as the product, and the main reason for this has been to allow differentiation of the node outputs with respect to the Gaussian (or other) parameters. ZHANG, HANG, TAN and WANG have shown in [7] that this is not necessary since a calculus can be developed for both the MIN and MAX operators, and indeed, combinations thereof. In the next section we give the essential results from their work for this type of network.

The outputs of the second layer are then truth values or rule firing strengths of each rule and these are propagated along weighted arcs forward to the next layer. This third layer is made up of MAX/MIN units, which first compute the minimum of the arc weights and the output of the relevant activating unit in the second layer. These weights may be chosen as the membership functions of the $N$ consequent fuzzy sets, defined at $M$ sample points in the appropriate universe of discourse, where $M$ is the number of MAX/MIN nodes. Thus the weight vector $\mathbf{w}_i = [w_{i,1}, w_{i,2}, \ldots, w_{i,M}]$ containing the weights on the arcs emanating from unit $i$, $i = 1, \ldots, N$ represents the consequent fuzzy set for rule $i$, as it would be given for inclusion in a rule base. (Note that our earlier remarks still apply in that unless a 'sum-to-one' convention is adopted for the definition of the antecedent fuzzy sets, then no 'consequent' set $\mathbf{w}_i$ would appear as the network output.) The calculation of the minimum of the output of unit $i$ in the second layer, and the weights on the arcs from it, is equivalent to clipping the antecedent set at the level set by the firing strength. The second part of the operation of these units is the computation of the maximum of all these clipped inputs. This means that with an identity transfer function, each unit in this layer provides as its output the value of the membership function of the union of the clipped fuzzy sets at a particular sample point $j$, $j = 1, \ldots, M$, in the output universe of discourse. Again in [7] it is shown how MAX/MIN functions can be differentiated (almost everywhere), and thus a trainable network can be designed.

As shown in *Fig. 5*, the output of the network is the consequent fuzzy set corresponding to the input crisp state. Additional layers can be added to perform the defuzzification process, for example using the centre of area method.

Clearly this network is capable of carrying out compositional inference using the individual rule firing approach, and in this case the antecedent and desired consequent fuzzy sets are loaded onto the network using

1. the location and radius of the Gaussian functions, or their equivalents, for the antecedent sets, and
2. the network weights for the consequent sets.

When this has been done, compositional inference is performed exhibiting the usual rule overlap phenomena. The key question is whether the network can be adapted from this configuration, or indeed trained from

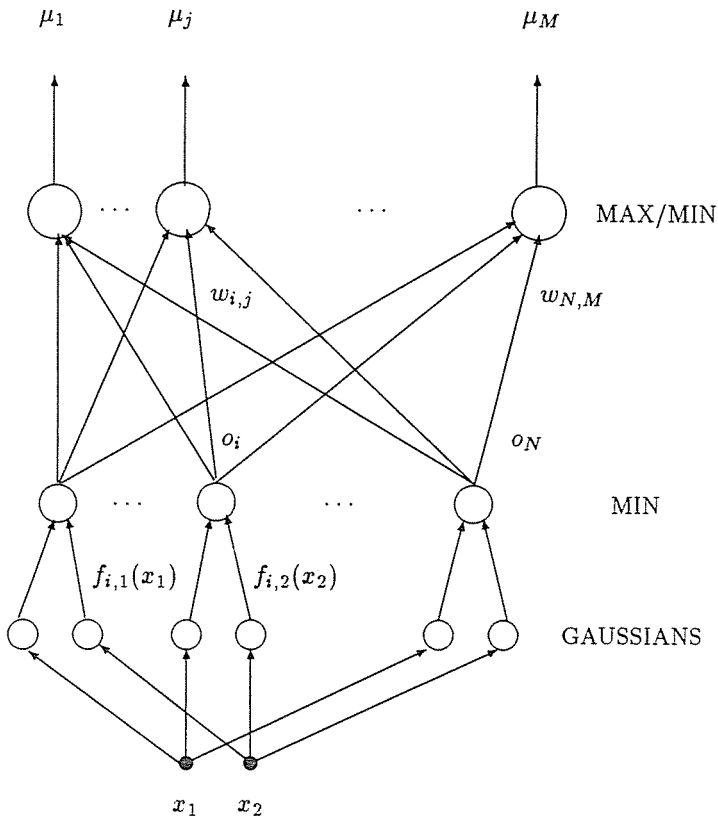


*Fig. 5.* The artificial Neural Inference network

system data with no preconceived (expert) rules given initially. This will only be possible, using gradient techniques, if the network output can be differentiated with respect to the Gaussian (or other) parameters, and the network weights. This will entail the differentiation of MAX/MIN functions, and we consider this in the next section.

## 8. Construction of a Training Algorithm

The development of an adaptive or training algorithm based on gradient descent depends upon two definitions of variants of the Heaviside step function. They are:

1.

$$\text{lor}\,(x) = \begin{cases} 1, & \text{if} \quad x > 0 \\ \frac{1}{2}, & \text{if} \quad x = 0 \\ 0, & \text{if} \quad x < 0 \end{cases} \tag{8}$$

2.

$$\operatorname{pos}(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \tag{9}$$

It then follows that we can write for any two functions $f(x)$, $g(x)$,

$$f(x) \vee g(x) = \operatorname{lor}[f(x) - g(x)]f(x) + [g(x) - f(x)]g(x)$$

and we note that if $f(x) = g(x)$, then $f(x) \vee g(x) = 1/2(f(x) + g(x))$.

This idea can be extended to a set of functions $F(x) = \{f_i(x),\ i = 1, \ldots, n\}$, and then we have

$$\bigvee f(x) = \frac{2^{\overline{K}(F)}}{\overline{K}(F)} \sum_{i=1}^{n} \left\{ \prod_{j=1}^{n} \operatorname{lor}[f_i(x) - f_j(x)] \right\} f_i(x)\,,$$

where

$$\overline{K}(F) \sum_{i=1}^{n} \prod_{j=1}^{n} \operatorname{poz}[f_i(x) - f_j(x)]\,. \tag{10}$$

With this definition, it follows that if at a particular point $x_r$, $\bigvee f(x_r) = f_j$, $j = 1, \ldots, r < n$, after re-ordering if necessary, then

$$\bigvee F(x_r) = 2^r \frac{1}{r} \sum_{j=1}^{r} \frac{f_j(x_r)}{2^r} = \frac{1}{r} \sum_{j=1}^{r} f_j(x_r)\,.$$

We also have for the minimum of two functions $f(x)$ and $g(x)$,

$$f(x) \wedge g(x) = \operatorname{lor}[g(x) - f(x)]f(x) + \operatorname{lor}[f(x) - g(x)]g(x)\,,$$

which can also be extended in a similar way to a set of functions. In order to develop a gradient descent based adaptive algorithm, the question of differentiating functions defined in this way has to be considered, and in [7] it is established that such functions are differentiable almost everywhere in $\mathbf{R}$. In particular, the following results hold.

1. If $a$ is a constant and $f(x)$ is differentiable at $x$, then

$$\frac{\mathrm{d}}{\mathrm{d}x} a \vee f(x) = \operatorname{lor}[f(x) - a] \frac{\mathrm{d}f(f)}{\mathrm{d}x}\,.$$

2. (a) If $f(x)$ and $g(x)$ are differentiable at $x$, then

$$\frac{\mathrm{d}}{\mathrm{d}x} f(x) \vee g(x) = \operatorname{lor}[f(x) - g(x)] \frac{\mathrm{d}f(x)}{\mathrm{d}x} + \operatorname{lor}[g(x) - f(x)] \frac{\mathrm{d}g(x)}{\mathrm{d}x}\,.$$

(b)

$$\frac{\mathrm{d}}{\mathrm{d}x} f(x) \wedge g(x) = \operatorname{lor}[g(x) - f(x)] \frac{\mathrm{d}f(x)}{\mathrm{d}x} + \operatorname{lor}[f(x) - g(x)] \frac{\mathrm{d}g(x)}{\mathrm{d}x} .$$

3. If the $n$ functions in the set $F = \{f_1(x), f_2(x), \ldots, f_n(x)\}$ are all continuously differentiable in $\mathbf{R}$, and if $\bigvee F$ is differentiable at $x$, then $2a$ generalises to

$$\frac{\mathrm{d}}{\mathrm{d}x} \bigvee F(x) = \frac{2^{\overline{K}(F)}}{\overline{K}(F)} \sum_{i=1}^{n} \left\{ \prod_{j=1}^{n} \operatorname{lor}[f_i(x) - f_j(x)] \right\} f_i'(x) ,$$

where $\overline{K}(F)$ is defined in (10).

It is also established in [7] that a training algorithm based on gradient descent for networks such as the ANI-net will converge to a local minimum with probability 1. In order to derive such a training algorithm, we use the network as shown in *Fig. 5*, that is, with $M$ outputs, $\mu_j$ $j = 1, \ldots, M$. The output $o_i$, $i = 1, \ldots, N$ of the $i$-th minimisation unit will be given by

$$o_i = f_{i,1}(x_1) \wedge f_{i,2}(x_2) \operatorname{lor}[f_{i,2}(x_2) - f_{i,1}(x_1)] f_{i,1}(x_1) +$$

$$+ \operatorname{lor}[f_{i,1}(x_1) - f_{i,2}(x_2)] f_{i,2}(x_2) ,$$

where

$$f_{i,k}(x_k) = \exp\left\{ -\frac{(x_k - c_{i,k})^2}{\sigma_{i,k}^2} \right\} \qquad k = 1, \ldots, n \tag{11}$$

and $c_{i,k}$ and $\sigma_{i,k}$ are adjustable parameters. The outputs $\mu_j$ are then generated as

$$\mu_j = \bigvee_{i=1}^{N} \{w_{i,j} \wedge o_i\} \qquad j = 1, \ldots, M .$$

In order to derive a gradient descent algorithm, after defining a suitable error measure $E$, say, we must calculate, *inter-alia* the partial derivatives $\frac{\partial \mu_j}{\partial w_{i,j}}$, $\frac{\partial \mu_j}{\partial o_i}$ and $\frac{\partial o_i}{\partial \alpha_{i,k}}$, where $\alpha_{i,k}$ is $c_{i,k}$ or $\sigma_{i,k}$. Consider first $\frac{\partial \mu_j}{\partial w_{i,j}}$ for $i = 1, \ldots, N$ and $j = 1, \ldots, M$. We have

$$\begin{aligned} \frac{\partial \mu_j}{\partial w_{i,j}} &= \frac{\partial}{\partial w_{i,j}} \bigvee_{i=1}^{N} \{w_{i,j} \wedge o_i\} \\ &= \frac{\partial}{\partial w_{i,j}} \left\{ (w_{i,j} \wedge o_i) \vee \bigvee_{\substack{i'=1 \\ i' \neq i}}^{N} \{w_{i',j} \wedge o_{i'}\} \right\} \end{aligned}$$

$$= \text{lor}\left[(w_{i,j} \wedge o_i) - \bigvee_{\substack{i'=1 \\ i' \neq i}}^{N} \{w_{i',j} \wedge o_{i'}\}\right] \frac{\partial}{\partial w_{i,j}}(w_{i,j} \wedge o_i)$$

$$= \text{lor}\left[(w_{i,j} \wedge o_i) - \bigvee_{\substack{i'=1 \\ i' \neq i}}^{N} \{w_{i',j} \wedge o_{i'}\}\right] \times \text{lor}\,[o_i - w_{i,j}]\,.$$

This result enables us to perform weight updating using an equation of the form

$$w_{i,j}^{\text{new}} = w_{i,j}^{\text{old}} - \theta_w \frac{\partial E}{\partial \mu_j} \frac{\partial \mu_j}{\partial w_{i,j}}\,, \tag{12}$$

where $\theta_w$ is a learning rate.

In a similar way we can show that

$$\frac{\partial \mu_j}{\partial o_i} = \text{lor}\left[(w_{i,j} \wedge o_i) - \bigvee_{\substack{i'=1 \\ i' \neq i}}^{N} \{w_{i',j} \wedge o_{i'}\}\right] \times \text{lor}\,[w_{i,j} - o_i]\,.$$

Now we have to calculate $\frac{\partial o_i}{\partial \alpha_{i,k}}$, where $\alpha_{i,k}$ is $c_{i,k}$ or $\sigma_{i,k}$. In the case when $n = 2$, it is easy to see that

$$\frac{\partial o_i}{\partial \alpha_{i,1}} = \text{lor}\,[f_{i,2}(x_2) - f_{i,1}(x_1)] \frac{\partial f_{i,1}}{\partial \alpha_{i,1}}\,,$$

where $\alpha_{i,1}$ is $c_{i,1}$ or $\sigma_{i,1}$, since $f_{i,2}(x_2)$ does not depend on either of these quantities. Similarly, we have

$$\frac{\partial o_i}{\partial \alpha_{i,2}} = \text{lor}\,[f_{i,1}(x_1) - f_{i,2}(x_2)] \frac{\partial f_{i,2}}{\partial \alpha_{i,2}}\,,$$

where $\alpha_{i,2}$ is $c_{i,2}$ or $\sigma_{i,2}$. The calculation of these derivatives is straightforward, using (11), and updating of the Gaussian parameters is by use of an equation similar in form to (12).

In our preliminary studies, we have succeeded in training an ANI-net, including added defuzzification layers, to represent a simple mapping, although the computational effort was significant. Nevertheless, it is satisfying to be able to take the relation between artificial neural networks and fuzzy inference systems one stage further.

## 9. Conclusions

In this paper, we have endeavoured to show the deep association which exists between Radial Basis Function artificial neural networks and the implementation of fuzzy logic. At the heart of this association is the Gaussian function, appearing as it does, in different rôles. By establishing the equivalence of a modified RBF network and a fuzzy inference system, we have proposed a method for rule extraction from (modified) RBF networks. Also, by examining the details of this equivalence, we see that there may be advantage in using as basis functions in such networks, functions which have (hyper)-elliptic rather than circular cross sections in 'pattern' space, as was anticipated by ALBRECHT and WERNER in 1966 [8]. The alternative view, that of needing to find an appropriate distance metric suggests that we consider further forms of generalisation for example, by the use of the Mahalanobis distance $D$ between two vectors $\mathbf{x}$ and $\mathbf{c}$, where

$$D^2 = (\mathbf{x} - \mathbf{c})^T \mathbf{S} (\mathbf{x} - \mathbf{c}) ,$$

and where $\mathbf{S}$ is a symmetric rather than a diagonal matrix.

The paper opened with a discussion of a network structure, designed to perform inference while avoiding the perceived shortcomings of the compositional rule of inference in terms of rule overlap. This led on to a consideration of a modified form RBF network which could model an adaptive fuzzy control system and the question of rule extraction was addressed. The approach which has been developed serves to show that the extraction of a single consequent fuzzy set for a given antecedent will not, in general, be possible. Naturally, this causes us to reflect on the meaning of the verbal forms of the rule base!

In the first form of RBF network considered, we paid attention to the need to produce 'genuine' membership function values as the outputs, that is with values in $[0, 1]$. In fact, for satisfactory operation, this is unnecessary, since subsequent de-fuzzification using the centre of area method, would yield the same value whether or not this had been done. Nevertheless, this was thought at the time to be appropriate to remain within the scope of fuzzy logic. A recent paper by MITAIM and KOSKO [9] might lead to second thoughts! There it is concluded that for the purpose of function approximation, the use of fuzzy sets defined by membership functions of the form $\operatorname{sinc} x = \frac{\sin x}{x}$ are the most efficient. In noting that $\operatorname{sinc} x$ can take negative values, the paper suggests that such values be interpreted as 'very low degrees of membership'!

The paper concluded with a discussion of our preliminary work on the ANI-net, and reported that the computational times seemed excessive for the task in hand. We are currently looking a little more closely at the mechanism for training, and in detail at making economies, for example, in using fuzzy error measures.

In our view, the most interesting aspect of this work has been the cross-domain insight which has been gained between two apparently distinct areas of 'soft computing'. As this field expands, it will be important to maintain this capability, in order to re-invent the wheel at regular intervals!

## References

[1] Advances in Numerical Analysis, Vol. 2, Oxford University Press, Oxford, 1992.
[2] BROOMHEAD, D. S. – LOWE, D. (1988): Multivariable Functional Interpolation and Adaptive Network, *Complex Systems*, Vol. 2, pp. 321–355.
[3] STEELE, N. C. – REEVES, C. R. – NICHOLAS, M. – KING, P. J. (1995): Radial Basis Function Artificial Networks for the Inference Process in Fuzzy Logic Based Control. *Computing*, Vol. 54, No. 2, pp. 99–117.
[4] ROGER JANG, J-S. – SUN, C-T. (1993): Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems. *IEEE Transactions on Neural Networks*, Vol. 4, No. 1, pp. 156–158.
[5] GODJEVAC, J. (1995): A Learning Procedure for a Fuzzy System: Application to Obstacle Avoidance, *Proc. ICSC Symposium on Fuzzy Logic*, Zurich, May 1995. ICSC Academic Press, 1995.
[6] GODJEVAC, J. (1996): Neuro-Fuzzy Controllers for Navigation of Mobile Robots. PhD Thesis, EPF-Lausanne, 1996.
[7] ZHANG, X. – HANG, C-C. – TAN, S. – WANG, P-Z. (1996): The Min-Max Function Differentiation and Training of Fuzzy Neural Networks, *IEEE Transactions on Neural Networks*, Vol. 7, No. 5, pp. 1139–1150.
[8] ALBRECHT, R. – WERNER, W. (1966): Ein Verfahren zur Identifizierung von Zeichen, deren Wiedergabe stationaeren statistischen Stoerungen unterworfen ist. *Computing*, Vol. 1, No. 1, pp. 1–7.
[9] MITAIM, S. – KOSKO, B. (1996): What is the Best Shape for a Fuzzy Set in Function Approximation? *Proceedings of Fuzz-IEEE 96*, New Orleans, Vol. 2, pp. 1237–1243.