1995

**Neuro-Fuzzy Systems for Intelligent Scientific Computation**

Narendran Ramakrishnan

Anupam Joshi

Sanjiva Weerawarana

Elias N. Houstis

*Purdue University, enh@cs.purdue.edu*

John R. Rice

*Purdue University, jrr@cs.purdue.edu*

**Report Number:**

95-026
NEURO-FUZZY SYSTEMS FOR INTELLIGENT SCIENTIFIC COMPUTATION

Narendran Ramakrishnan
Anupam Joshi
Sanjiva Weerawarana
Elias N. Houstis
John R. Rice

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD-TR-95-026
April 1995
Neuro-Fuzzy Systems for Intelligent Scientific Computation *

Narendran Ramakrishnan, Anupam Joshi, Sanjiva Weerawarana,
Elias N. Houstis and John R. Rice
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907, USA

Abstract
Intelligence has been envisioned as a key component of future problem solving environments for scientific computing. This paper describes a computationally intelligent approach to address a major problem in scientific computation i.e., the efficient solution of partial differential equations (PDEs). This approach is implemented in PYTHIA - a system that supports smart parallel PDE solvers. PYTHIA provides advice on what method and parameters to use for the solution of a specific PDE problem. It achieves this by comparing the characteristics of the given PDE with those of previously observed classes of PDEs. An important step in the reasoning mechanism of PYTHIA is the categorization of PDE problems into classes based on their characteristics. Exemplar based reasoning systems and backpropagation style neural networks have been earlier used to this end. In this paper, we describe the use of fuzzy min-max neural networks to realize the same objective. This method converges faster, is more accurate, generalizes very well and provides on-line adaptation. This technique makes certain assumptions about the pattern classes underlying the domain. In applying the fuzzy min-max network to our domain, we improve the method by relaxing these assumptions. This scheme will form a major component of future problem solving environments for scientific computing that are being developed by our group.

1 Introduction

It has been envisioned that future problem solving environments (PSEs) will have at least some form of intelligence and will provide a natural interface within well defined domains of scientific application [6]. In this paper, we address how such intelligence can be achieved by a combination of neural and fuzzy mechanisms. We describe the method and apply it to a major problem in scientific computation i.e., the efficient numerical solution of partial differential equations (PDEs). This depends on many factors including the nature of the operator, the mathematical behavior of the coefficients and the exact solution of the PDE, the type of boundary and initial conditions, and the geometry of the space domains of definition.

There have been numerous systems that have been proposed for assisting in various aspects of the PDE solution process. An abstract model for the algorithm selection problem is described by Rice [9]. An experimental methodology implementing this model has also been developed in [10]. In [7], Moore et al. describe a strategy for determining a geometry discretization that leads to a solution guaranteed to be within a certain prescribed accuracy. At the other end of the PDE solution process, expert systems have been designed that apply self-validating methods in an economical manner to systems of linear equations. These help to guide the internals of a linear system solver. In [2], Dyksen and Gritter describe an expert system for selecting solution methods for elliptic PDEs based on problem characteristics. Weerawarana [14] argues that using problem characteristics is not sufficient and proposes an exemplar based reasoning

*This work was supported in part by NSF awards ASC 9404859 and CCR 9202536, AFSOR award F49620-92-J-0069 and ARPA ARO award DAAH04-94-G-0010
system that uses performance profiles of PDE solvers to determine a solver for a particular PDE problem. Joshi et al. [1] describe connectionist schemes for the problem of classifying PDE problems into classes based on properties of their solutions.

This paper deals mainly with the PYTHIA system. PYTHIA is an advisory system that supports smart parallel PDE solvers for partial differential equations. It provides valuable advice on what method and parameters to use for the solution of a specific PDE problem. An important step in this reasoning process is the categorization of PDE problems into classes based on their characteristics. We have developed a hybrid neuro-fuzzy methodology around fuzzy min-max neural networks and applied it to this particular classification problem. The method is quite general and can be applied to any pattern classification problem.

The rest of the paper is organized as follows: Section 2 introduces the theory of fuzzy min-max neural networks. Section 3 discusses the shortcomings of the method that render it unsuitable for application to certain problem domains. An enhanced method and its mechanism of operation are detailed in Section 4. A performance evaluation of the method is carried out in Section 5 that applies the algorithm to the classes determination problem in PYTHIA. Section 6 looks at interesting variations of the scheme and suggests application areas where they might come useful. Section 7 concludes by summarizing and provides pointers for further research in this field.

2 Fuzzy Min Max Neural Networks

Fuzzy Min-Max neural networks were proposed by Simpson [12] as a supervised learning paradigm that finds reasonable decision boundaries in pattern space. We now briefly describe Simpson’s method. This method uses fuzzy sets to describe pattern classes. Each fuzzy set is the fuzzy union of several n-dimensional hyperboxes. Such hyperboxes define a region in n-dimensional pattern space that have patterns with full class membership. A hyperbox is completely defined by its min-point and max-point. It has associated with it a fuzzy membership function and a scalar denoting the class it corresponds to in pattern space. This function helps to view the hyperbox as a fuzzy set and such “hyperbox fuzzy sets” can be aggregated to form a single fuzzy set class. The advantage of this approach is that it provides a degree-of-membership information that is useful in decision making. Without any loss of generality, the pattern space is considered to be the n-dimensional unit hypercube \( I^n \) and the membership values, in addition, are taken to be in the range \([0,1]\). Each hyperbox fuzzy set \( B_j \) can then be described as the 3-tuple

\[
B_j = \{V_j, W_j, c\}
\]  

where \( V_j \) denotes the min-point in n-space, \( W_j \) represents the max point in n-space and \( c \) denotes the class to which the fuzzy set \( B_j \) corresponds. (It is worth noting that each hyperbox fuzzy set is created at the instance of observation of a pattern so that the class information is known while the fuzzy set is initialized.)

The membership function \( m_j \) for the \( j^{th} \) hyperbox obeys the expression:

\[
0 \leq m_j(A_k) \leq 1
\]

i.e., \( m_j \) measures the degree to which the \( k^{th} \) pattern \( A_k \) falls inside the \( j^{th} \) hyperbox. Simpson chooses this function to be,

\[
m_j(A_k) = 1 - \frac{1}{n} \sum_{i=1}^{n} [f(a_{ki} - w_{ji}) + f(v_{ji} - a_{ki})]
\]  

\[
where f(x) = \max(0, \min(\gamma x, 1))
\]  

The above expression can be interpreted as follows: \( f(a_{ki} - w_{ji}) \) measures the degree to which \( A_k \) falls above the max-point \( W_j \) for the \( i^{th} \) dimension while \( f(v_{ji} - a_{ki}) \) measures the degree to which it falls below the min-point \( V_j \). Thus their sum denotes the degree to which the point falls outside the hyperbox \( B_j \). When this sum is deducted from 1, we get the degree to which it falls inside the hyperbox. \( f(x) \) is a squashing function that clamps its argument to lie in the range \([0,1]\). The sensitivity parameter \( \gamma \) represents the rate at which the membership function varies as the distance between \( A_k \) and \( B_j \) varies.
Learning in the fuzzy min-max network proceeds by placing and adjusting the hyperboxes in pattern space. Using the terminology of equation (1), the fuzzy set that describes a pattern class can be represented by an aggregate (the fuzzy union) of several fuzzy sets i.e.,

\[ P_i = \bigcup_{j \in I} B_j \]  

(3)

where \( I \) is the index set of all hyperboxes associated with class \( i \). The learning is an expansion or contraction process. The training set consists of a set of \( N \) ordered pairs \( \{ A_h, d_h \} \), where \( A_h = (a_{h1}, a_{h2}, ..., a_{hn}) \in \mathbb{R}^n \) is the input pattern and \( d_h \in \{1, 2, ..., p\} \) is the index of one of the \( p \) classes. The learning algorithm operates by selecting an input pattern and finding a hyperbox for this pattern’s class to include it. If needed, the hyperbox is expanded suitably to include it. If such a hyperbox cannot be found, a new hyperbox is created. This procedure allows new classes to be added without retraining and also refines existing classes over time. However, this might result in hyperbox overlap. If such an overlap occurs between hyperboxes of different classes, then the overlap is eliminated by a contraction process. In other words, the algorithm eliminates overlap between hyperboxes representing different classes but allows hyperboxes of the same class to overlap. The only parameter that needs to be tuned is the maximum hyperbox size \( \theta \) i.e., the size beyond which a hyperbox cannot be expanded. When this value is set to zero, the algorithm described above reverts to the k-nearest-neighbor classifier algorithm [3].

Recall in the network consists of calculating the fuzzy union of the membership function values produced from each of the fuzzy set hyperboxes. This can be implemented as a three-layer neural network (Fig.2.1). The input layer has \( n \) processing elements, one for each dimension of the input pattern \( A_h \). The hidden layer has \( m \) elements, one for each of the hyperbox fuzzy sets that are formed. There are two sets of connections from each input node to each of the \( m \) hyperbox fuzzy set nodes - the min vector and the max vector. These connections are adjusted by the learning algorithm. The transfer function for these nodes is the membership function defined in (2) above. The output layer has \( p \) processing elements, one for each class in the pattern space. The connections between the hidden layer and the output layer are binary valued and are given by

\[ u_{ji} = \begin{cases} 1, & \text{if } b_j \text{ is a hyperbox for class } c_i \\ 0, & \text{otherwise} \end{cases} \]  

(4)

where \( b_j \) is the \( j^{th} \) node in the hidden layer and \( c_i \) is the \( i^{th} \) node in the output layer. The output of the \( i^{th} \) node output layer is given by

\[ c_i = \max_{j=1}^{m} b_j u_{ji} \]  

(5)

3 Limitations of the method

Simpson’s method assumes that the pattern classes underlying the domain are mutually exclusive. It may be noted that the types of pattern classes that characterize problems in many real world domains are frequently not mutually exclusive. For example, some PDEs might have an analytic solution, some might have mixed boundary conditions, but some PDEs can be both analytic and have mixed boundary conditions.

Simpson’s algorithm would fail to account for a situation where one problem might be expected to belong to several classes. Also, the only parameter in the original method was the maximum hyperbox size parameter \( \theta \). It is not reasonable to assume that one parameter is sufficient to tune the entire system. Moreover, the effect of \( \theta \) on classification accuracy was not completely understood.

Interestingly, these same restrictions have been characteristic of another classical neural network paradigm - the fuzzy adaptive resonance theory (fuzzy ART) of Carpenter et.al. [5]. This is an analog pattern clustering system that combines the concepts of fuzzy logic with the original ART networks created in [4].

An unsupervised version of Simpson’s algorithm has also been proposed that clusters unlabeled pattern data into hyperboxes [13]. Again, there is an assumption that the clusters to be found are mutually exclusive.
main()
{
    for each set of labeled data \( i \) from training set do
        for each class \( j \) that pattern \( i \) belongs to
            box = identifyexpandablebox();
            if (box = NOTAVAILABLE) addnewbox();
            else expandbox(box);
            flag = checkforoverlap();
            if (flag = true) contracthyperboxes();
}

Figure 1: The modified algorithm

In the next section, we describe an enhanced method that operates with overlapping and non-exclusive classes. In this process, we introduce another parameter \( \delta \) to tune the system. We also study the effect of the parameters \( \theta \) and \( \delta \) on classification accuracy by applying the method to a real-world problem in scientific computation.

4 The modified algorithm

The new fuzzy min-max classification algorithm is described in Fig. 4 above.

Consider the \( k \)th ordered pair from the training set. Let the desired output for the \( k \)th pattern be \([1,1,0,0,...,0]\). The algorithm above considers this as two ordered pairs containing the same pattern but have training outputs - \([1,0,0,0,...,0]\), \([0,1,0,0,...,0]\) respectively. In other words, the pattern will be associated with both class 1 and class 2. But according to the algorithm, one data point can have complete membership in only hyperboxes of the same class.

It can be reasoned then that any algorithm for classifying data of this type should ensure that the pattern does not belong to both classes to full extent. The above procedure results in the pattern having equal degrees of membership in both the hyperboxes but is not completely contained in either of them. Assume that the network is first trained with the desired output as \([1,0,0,0,...,0]\). This results in the \( k \)th pattern having complete containment in a hyperbox of class 1 (because the 1st bit is set to 1). Then when we train the same pattern with \([0,1,0,0,...,0]\), a hyperbox of class 2 will be created/expanded to include the \( k \)th pattern. This will result in hyperbox overlap. The hyperbox contraction step detailed below ensures that both the hyperboxes are adjusted so that each of them contain the \( k \)th pattern to the same degree (which will be less than 1).

(a) Hyperbox Expansion: Given labeled data of the form \( \{A_h, d_h\} \), find the hyperbox \( b_j \) that represents the same class as \( d_h \), provides the highest degree-of-membership and allows expansion (if needed). Since we bound the maximum hyperbox size by \( \theta \), the following condition is satisfied:

\[
\frac{1}{n} \sum_{i=1}^{n} (\max(w_{ji}, a_{hi}) - \min(w_{ji}, a_{hi})) \leq \theta \tag{6}
\]

Then, the min-points and the max-points are adjusted by the equations:

\[
w_{ji}^{(k+1)} = \min(w_{ji}^{(k)}, a_{hi}) \forall i = 1, 2, \ldots, n \tag{7a}
\]

\[
w_{ji}^{(k+1)} = \max(w_{ji}^{(k)}, a_{hi}) \forall i = 1, 2, \ldots, n \tag{7b}
\]
Fig. 2.1
Implementation of the fuzzy min-max neural network
(b) Overlap Testing: A dimension-by-dimension comparison between hyperboxes is effected here. This test is conducted between the hyperbox expanded in the previous step and any other hyperbox that represents a different class. Let $B_i$ be the one expanded in the previous step and $B_j$ represent another hyperbox of a different class. If at least one of the following conditions is satisfied for a dimension, then we conclude that overlap exists between the hyperboxes. $\Delta^{(k)}$ is initialized to 1. (The figures below indicate a two-dimensional case where overlap has been detected along the first dimension i.e., along the abscissa). The various conditions to be tested for are as follows:

Condition 1 (Fig. 4.1): $v_{ji} < v_{li} < w_{jj} < w_{lj}$

\[ \Delta^{(k+1)} = \min(w_{jj} - v_{li}, \Delta^{(k)}) \] (8a)

Condition 2 (Fig. 4.2): $v_{li} < v_{ji} < w_{li} < w_{ji}$

\[ \Delta^{(k+1)} = \min(w_{li} - v_{ji}, \Delta^{(k)}) \] (8b)

Condition 3 (Fig. 4.3): $v_{ji} < v_{li} < w_{li} < w_{ji}$

\[ \Delta^{(k+1)} = \min(w_{li} - v_{ji}, w_{ji} - v_{li}, \Delta^{(k)}) \] (8c)

Condition 4 (Fig. 4.4): $v_{li} < v_{ji} < w_{li} < w_{ji}$

\[ \Delta^{(k+1)} = \min(w_{ji} - v_{li}, w_{li} - v_{ji}, \Delta^{(k)}) \] (8d)

If $\Delta^{(k+1)} > \Delta^{(k)}$, then there was no overlap and the next contraction step is unnecessary. If, on the other hand, $\Delta^{(k+1)} < \Delta^{(k)}$, then overlap has occurred in the $i^{th}$ dimension and the $(i + 1)^{th}$ dimension is now checked for overlap after setting $\Delta^{(k)} = \Delta^{(k+1)}$.

(c) Hyperbox Contraction: If overlap was detected in the $i^{th}$ dimension, as detailed above, we minimally adjust the $i^{th}$ dimensions of each of the overlapping hyperboxes. In other words, we try to adjust the hyperboxes so that only one of the min/max points is altered at a time. We examine the same four cases as above,

Condition 1 (Fig. 4.1): $v_{ji} < v_{li} < w_{ji} < w_{li}$

\[ w_{ji}^{(k+1)} = v_{ji}^{(k+1)} = \frac{w_{ji}^{(k)} + v_{ji}^{(k)}}{2} \] (9a)

Condition 2 (Fig. 4.2): $v_{li} < v_{ji} < w_{li} < w_{ji}$

\[ w_{li}^{(k+1)} = v_{li}^{(k+1)} = \frac{w_{li}^{(k)} + v_{li}^{(k)}}{2} \] (9b)

Condition 3a (Fig. 4.3(a)): $v_{ji} < v_{li} < w_{li} < w_{ji}$ and $(w_{li} - v_{ji}) < (w_{ji} - v_{li})$

\[ v_{ji}^{(k+1)} = w_{ji}^{(k)} \] (9c1)

Condition 3b (Fig. 4.3(b)): $v_{ji} < v_{li} < w_{li} < w_{ji}$ and $(w_{li} - v_{ji}) > (w_{ji} - v_{li})$

\[ v_{ji}^{(k+1)} = v_{li}^{(k)} \] (9c2)

Condition 4a (Fig. 4.4(a)): $v_{li} < v_{ji} < w_{li} < w_{ji}$ and $(w_{li} - v_{ji}) < (w_{ji} - v_{li})$

\[ w_{li}^{(k+1)} = v_{li}^{(k)} \] (9d1)

Condition 4b (Fig. 4.4(b)): $v_{li} < v_{ji} < w_{li} < w_{ji}$ and $(w_{li} - v_{ji}) > (w_{ji} - v_{li})$

\[ v_{li}^{(k+1)} = v_{ji}^{(k)} \] (9d2)
Fig. 4.3(a)

Fig. 4.3(b)
Fig. 4.4(a)

Fig. 4.4(b)
Since each pattern can belong to more than one class, we need to define a new way to interpret the output of the fuzzy min-max neural network. In the original algorithm, we locate the node in the output layer with the highest value and set the corresponding bit to 1. All other bits are set to zero. In this way, a hard decision is obtained.

In the modified algorithm, however, we introduce a parameter $\delta$ and we set to 1 not only the node with the highest output but also the nodes whose outputs fall within a band $\pm\delta$ of the output value. This results in more than one output node getting included and consequently, aids in the determination of non-exclusive classes.

5 Experimental results

In this section, we study the effectiveness of fuzzy min-max neural networks by applying them to the problem of categorization of a given PDE problem into one of several classes. (As the classes were non-exclusive, the original network could not be applied to the domain.) The following non-exclusive classes were defined in PYTHIA (the number of exemplars in each class is given in parentheses):

(i) SINGULAR: PDE problems whose solutions have at least one singularity (6).
(ii) ANALYTIC: PDE problems whose solutions are analytic (35).
(iii) OSCILLATORY: PDE problems whose solutions oscillate (34).
(iv) BOUNDARY-LAYER: Problems that depict a boundary layer in their solutions (32).
(v) BOUNDARY-CONDITIONS-MIXED: Problems that have mixed boundary conditions (74).

Each PDE problem was coded as a 32-component characteristic vector and the number of classes were 5. Since the above algorithm works for labeled data of the form $\{A_h, d_h\}$, there is an implicit assumption that each pattern should belong to at least one class. However, in the problem domain, we may come across instances of PDEs that do not belong to any of the above defined classes. To circumvent this difficulty, we define a sixth class as follows:

(vi) SPECIAL: PDE Problems whose solutions do not fall into any of the classes (i)-(v).

There were a total of 167 problems in the PDE population that belong to at least one of these classes. This data was split into two parts - one part containing two thirds of the exemplars and the other part was used to test the generalization of the network. The following set of experiments were conducted:

(i) Effect of $\theta$: In this set of experiments, the max. hyperbox size was varied continuously and its effect on other variables were studied. In particular, it can be observed that when $\theta$ was increased, a lesser number of hyperboxes needed to be formed, i.e., when $\theta$ tends to 1, the number of hyperboxes formed is 6 - the number of classes in the domain (Fig. 2). Also performance on the training set and the test set steadily improved as $\theta$ was decreased (Fig. 3). Performance on the training set was, expectedly, better than that on the test set. An optimal error was achieved at a $\theta$ value of 0.00125. When $\theta > 0.00125$, the error increased on both the sets and when $\theta < 0.00125$, the network overfit the training data so that its performance on the test set started to decline. The number of hyperboxes formed for this optimal value of $\theta$ was 82 - approximately double the size of the dimension of the pattern space.

(ii) Effect of $\delta$: In this experiment, we set $\theta = 0.00125$ (the optimal value) and we vary the threshold $\delta$ by assigning to it the values 0.01, 0.02, 0.05 and 0.09. It is observed that when $\delta$ was increased, more output nodes tend to get included in the "reading-off" stage so that the overall error increased. The optimal error mentioned in Expt.(i) was achieved at a $\delta = 0.01$. Figs. 4, 5, 6, 7 show scatter plots of results for these values of $\delta$. 

6
Figure 2: The number of hyperboxes formed expressed as a function of $\theta$ - the maximum hyperbox size.

Figure 3: Effect of $\theta$ on the performance. The solid line indicates the error on the training set while the dashed line indicates the error on the test set.
Figure 4: Scatter plot of results for optimum $\theta$ and $\delta = 0.01$.

Figure 5: Scatter plot of results for optimum $\theta$ and $\delta = 0.02$. 
Figure 6: Scatter plot of results for optimum $\theta$ and $\delta = 0.05$.

Figure 7: Scatter plot of results for optimum $\theta$ and $\delta = 0.09$. 
(iii) **On-line adaptation**: The last series of experiments conducted were to test the fuzzy min-max neural network for its on-line adaptation i.e., each pattern was incrementally presented to the network and the error on both sets was recorded at each stage. We can observe from Fig. 8 that the number of hyperboxes formed slowly increase from 1 to the optimal number in Expt.(i) - 62. Also, performance on both sets steadily improved to the values obtained in Expt.(i) (Fig. 9).

6 **Future Directions**

It can be observed that the hyperboxes formed by the proposed algorithm are isothetic (i.e., they have their sides parallel to one of the orthogonal axes). We inherit this feature from Simpson’s original work. For some types of pattern data, the hyperboxes formed do not give a realistic idea of the classification boundaries. A 2-D example is illustrated in Fig. 6.1 where an isothetic hyperbox does not do a very good job of modeling the system. Thus, an optimal hyperbox for such data might be found to be inclined to the axes by some angle. In ongoing work, we are extending this algorithm to vary not just the size of the hyperbox, but it’s orientation as well.

Also, for data of the form in Fig. 6.2, the natural pattern classification geometry will be that of a hypersphere. Thus, if we were to use hyperspheres/hyperellipsoids as space-covering primitives, a better classification can be obtained. Some work in this direction has been done in [11] and [8].

Our ongoing research addresses the issue of finding an algorithm that covers the pattern space with appropriate primitives. These questions can be related to general problems in n-dimensional computational geometry. Their exact solution is either intractable, or exceedingly expensive computationally. We are working on developing fast heuristics that will allow us to improve the performance of the present fuzzy min-max system.
7 Conclusion

In this paper, we have developed a fuzzy min-max approach to pattern classification. It has considerable advantages over the conventional scheme which could not cater for mutually non-exclusive classes. The effectiveness of the method was demonstrated by application to an important problem pertaining to partial differential equations. This method is very accurate, provides good convergence, has excellent generalization abilities and is well-suited for on-line adaptation. It is proposed that this scheme will form a major component of future problem solving environments for partial differential equations and, in general, intelligent scientific computing. Future directions include borrowing ideas from computational geometry to allow non-isothetic hyperboxes and in general, use hyperspheres or hyperellipsoids as our pattern space covering primitives.

References


