An Approach to Rule-Based Knowledge Extraction

Yaochu Jin[§,¶], Werner von Seelen[§] and Bernhard Sendhoff[§]
[§]Institut für Neuroinformatik, Ruhr-Universität Bochum
D-44780 Bochum, Germany
[¶]Electrical Engineering Department, Zhejiang University
Hangzhou 310027, P. R. China

Abstract

The extraction of easily interpretable knowledge from the large amount of data measured in experiments is well desirable. This paper proposes a method to achieve this. A fuzzy rule system is first generated and optimized using evolution strategies. This fuzzy system is then converted to an RBF neural network to refine the obtained knowledge. In order to extract understandable fuzzy rules from the trained RBF network, a neural network regularization technique called adaptive weight sharing is developed. Simulation results on the Mackey-Glass system show that the proposed approach to knowledge extraction is effective and practical.

1. Introduction

Artificial neural networks and fuzzy systems are two of the main modeling tools in soft computing. The key feature of neural networks is their learning capability, whereas in the case of fuzzy systems it is the comprehensible representation of knowledge. The nature of this representation allows prior expert knowledge to be incorporated into fuzzy systems. Thus, fuzzy systems are prefered if it is necessary that the knowledge is represented in an intelligible way. However, the adaptation ability of conventional fuzzy systems is often weak. To solve this problem, neural and evolutionary fuzzy systems [2, 5] have been proposed. Although such hybrid fuzzy systems have exhibited remarkable learning abilities, they usually lose their interpretability, which is the most essential feature of fuzzy systems.

Interpretability of a fuzzy system usually involves the following aspects. Firstly, the fuzzy partitioning for each input variable of the fuzzy system should be complete and different fuzzy subsets in a fuzzy partitioning should be well distinguishable. One direct method to achieve this is to limit the range of the parameters of membership functions during learning [8]. This can be achieved more flexibly with the help of a fuzzy similarity measure [4]. Secondly, the number of fuzzy subsets in a fuzzy partitioning should be limited and each fuzzy subset should have one unique membership function, to which a proper physical meaning can be assigned. Thirdly, fuzzy rules in the rule base should be consistent. Traditionally, this means that fuzzy rules with the same premise should have the same consequent [11, 3]. This has been extended in [4], where we have argued that rules with similar premises should have similar consequents. Finally, the number of rules in a fuzzy system should be as small as possible.

This paper aims at extracting interpretable fuzzy rules from data. A fuzzy system is first generated by virtue of evolution strategies. Then we convert the fuzzy system to an RBF neural network for further training to refine the aquired knowledge. After this learning stage, the RBF neural network can not be directly converted back to a clearly interpretable fuzzy system [12] because there may be numerous fuzzy subsets in a fuzzy partitioning which are hard to distinguish and hard to assign proper linguistic values to. To solve this problem, a network regularization algorithm called adaptive weight sharing is developed to train the RBF neural network further so that some of the basis functions as well as the output weights in the RBF network share certain values. As a result, a well interpretable fuzzy system can again be obtained. This method has proved to be successful by simulation studies on the Mackey-Glass time series.

2. Fuzzy System Generation and Optimization Using Evolution Strategies (ES)

In recent years, data based fuzzy rule generation and optimization have become popular. Several methods based on genetic algorithms or evolution strategies have been proposed. However, problems such as completeness and consistency of the fuzzy rule systems may occur. To deal with these problems, completeness and consistency indices have been proposed and incorporated into the cost function of the evolutionary algorithm [4]. The following subsections describe how complete and consistent fuzzy systems can be generated.

2.1. Completeness conditions

Suppose an input variable of a fuzzy system x is partitioned into M fuzzy subsets represented by A_1 **(x),** A_2 **(x),** .... A_M (x) on the universe of discourse U, then the partitioning is complete if the following condition holds:

$$\forall_{x\in U} \exists_{1\leq i\leq M} A_i(x) > 0 \tag{1}$$

In the optimization of fuzzy systems based on evolutionary algorithms or neural networks, it is often the case that either the fuzzy partitionings are incomplete or different fuzzy subsets in a fuzzy partitioning lack good distinguishability. To avoid this, we require that every two neighbouring fuzzy sets should satisfy the following constraints:

$$\delta_1 \leq S(A_i, A_{i+1}) \leq \delta_2 \tag{2}$$

where $S(A_i, A_{i+1})$ is called the fuzzy similarity measure between the two fuzzy subsets A_i and A_{i+1}, δ_1 and δ_2 are two thresholds of the fuzzy similarity measure, where δ_1 should be greater than zero to keep the fuzzy partitioning complete and δ_2 should be sufficiently smaller than 1 to ensure good distinguishability. The fuzzy similarity measure is defined by:

$$S(A_i, A_{i+1}) = \frac{M(A_i \cap A_{i+1})}{M(A_i) + M(A_{i+1}) - M(A_i \cap A_{i+1})} \tag{3}$$

where *M(.)* is called the size of the fuzzy set. If fuzzy set ***A(x)*** has a Gaussian membership function with center μ and width (or variance) σ, then *M (A(x))* can be calculated as:

$$\boldsymbol{M(A(x))} = \int_{-\infty}^{+\infty} \exp\left(-\frac{(x-\mu)^2}{\sigma^2}\right) dx \tag{4}$$

It is noticed that if $S(A_i, A_{i+1})$ equals 1, the two fuzzy sets overlap completely, i.e. A_i and A_{i+1} are equal. On the other hand, they do not overlap if $S(A_i, A_{i+1}) = 0$.

2.2. Consistency of fuzzy systems

Another important issue in data based fuzzy system generation is the consistency among the fuzzy rules and the consistency with the prior knowledge. It is easy to imagine that two fuzzy rules are inconsistent if they have the same if-part but different then-parts. However, we argue that two fuzzy rules may also be inconsistent even if their if-parts are different. To evaluate the consistency of two arbitrary fuzzy rules, definitions of *Similarity of Rule Premise* (SRP) and *Similarity of Rule Consequent* (SRC) are given. Consider the following two fuzzy rules:

R_i : If x_1 is $A_{i1}(x_1)$ and x_2 is $A_{i2}(x_2)$ and . . . and x_n is $A_{in}(x_n)$, then y is $B_i(y)$
R_k : If x_1 is $A_{k1}(x_1)$ and x_2 is $A_{k2}(x_2)$ and . . . and x_n is $A_{kn}(x_n)$, then y is $B_k(y)$

Then SRP and SRC between rule i and rule k are defined in terms of the fuzzy similarity measure as follows:

$$SRP(i,k) = \wedge_{j=1}^{n} S(A_{ij}, A_{kj}) \tag{5}$$

$$SRC(i,k) = S(B_i, B_k) \tag{6}$$

where n is the total number of the input variables. The consistency between rule *R(i)* and *R(k)* can now be defined as:

$$Cons(R(i), R(k)) = \exp\left\{ -\frac{\left(\frac{SRP(i,k)}{SRC(i,k)} - 1.0\right)^2}{\left(\frac{1}{SRP(i,k)}\right)^2} \right. \tag{7}$$

Thus, an inconsistency index of the fuzzy system can be given by:

$$f_{Incons} = \sum_{i=1}^{N} \sum_{\substack{1\leq k\leq N \\ k\neq i}} [1.0 - Cons(R(i), R(k))] \tag{8}$$

where N is the total number of rules. It is also feasible to evaluate the inconsistency between the fuzzy system and the prior knowledge expressed in fuzzy rules using this index.

Some remarks are to be made on the completeness and consistency conditions. Since they impose additional restrictions in generating fuzzy systems, they could be treated as a means of regularization. They play an important role if the training data are insufficient [4], irregular or noisy. On the other hand, they do not influence the rule evaluation significantly if the training data are sufficient, equally distributed and pure. In addition, the consistency condition makes it possible to avail of prior knowledge in terms of negative examples in fuzzy rule generation.

2.3. ES based fuzzy rule generation and optimization

We use a hybrid evolution strategy which is able to deal with optimization problems with both real and integer parameters [1] to generate and optimize an initial fuzzy rule system. The quality of the fuzzy system can be evaluated by the following cost function:

$$f = f_E + \xi \,.\, f_{Incons} + f_{Incomp} \tag{9}$$

where, f_E denotes the conventional error term, f_{Incons} is defined as in equation (8) and f_{Incomp} is a penalty term depending on whether the completeness condition in equation (2) is satisfied. If the condition is satisfied, $f_{Incomp} = 0$;

Otherwise, f_{Incomp} is so large that the corresponding individual can no longer survive.

The coding of the rule structure is as follows. Suppose there are n input variables and each input $x_i\,(i = 1, 2, \dots, n)$ has at most M_i subsets, then the rule base has at most $N = M_1 \times M_2 \times \dots \times M_n$ fuzzy rules. Therefore, the rule structure can be encoded by the following integer matrix:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{N1} & a_{N2} & \cdot & a_{Nn} & b_N \end{bmatrix}_{N \times (n+1)} \tag{10}$$

where $a_{ji} \in \{0, 1, 2, \dots, M_i\}$ and $b_j \in \{1, 2, \dots, M_{n+1}\}$, where M_{n+1} is the largest number of fuzzy subsets for the output. The integers $1, 2, \dots, M_i$ and $1, 2, \dots, M_{n+1}$ represent the corresponding fuzzy subsets of the inputs and the output respectively. Notice that a_{ji} may also equal zero, which implies that the i-th input does not appear in the j-th rule. If all the inputs do not appear, this rule is deleted automatically. Since we try to optimize the rule parameters as well as the rule structure, all the parameters of the membership functions will also be encoded. In this way, both the rule parameters and the rule structure will be evolved according to equation (9) by the evolution strategies and an initial fuzzy rule system will be generated.

3. Conversion of the Fuzzy System to an RBF Network

The significance of converting fuzzy systems to neural networks lies in two aspects. On the one hand, a fuzzy system can be refined taking advantage of the learning ability of neural networks. On the other hand, the structure of a neural network can be determined and prior knowledge can be incorporated into the network with the help of a fuzzy system. Moreover, we hope a trained RBF neural network can again be converted back to a fuzzy system. Although Jang and Sun [2] have proposed some restrictions for the equivalence between RBF neural networks and fuzzy systems, we think it is meaningful to look at this problem closer. We conclude that an interpretable fuzzy system and an RBF neural network are equivalent if the following conditions hold:

1. Both the fuzzy system and the neural network have Gaussian basis functions.
2. The number of fuzzy rules is equal to the number of receptive field units (or hidden nodes) in the RBF network.
3. The fuzzy system is either a zero-order Takagi-Sugeno model or a Mamdani model. If a Mamdani model is used, the corresponding defuzzification method should be the simplified weighted average [4]. In both cases, the product T-norm should be used.
4. The output of the RBF neural network should be normalized [9].
5. The receptive field units in the RBF network are allowed to have different variances.
6. Centers and variances from different receptive field units but for the same input variable should share certain values, which could construct a complete and well distinguishable fuzzy partitioning. If Mamdani fuzzy rules are expected, some of the output weights should also share.

In this paper, "weights" refer to both the parameters of the basis functions and the output weights. We think condition 4 is important in that it enables the Mamdani fuzzy systems as well as Sugeno fuzzy systems to be converted to an RBF neural network. Condition 5 avoids a dilemma, where either all of the membership functions of the fuzzy system are forced to have the same variance, or each membership function can be used only once. In the former case, the flexibility of the fuzzy system will be seriously harmed, while in the later case, the number of fuzzy subsets equal the number of fuzzy rules, which will obviously damage the interpretability of the fuzzy system, because the rule number is usually large. Finally, the weight sharing condition is necessary if a fuzzy system extracted from an RBF neural network is expected to be interpretable. That is to say, the effective number of parameters in an interpretable fuzzy system is smaller than the effective number of the parameters in its mathematically equivalent RBF network.

In this section, we convert the fuzzy system generated by evolution strategies to an RBF network for further training using the learning algorithm of the neural network. The final input-output relationship of the fuzzy system with n inputs and one output is expressed as follows:

$$y = \frac{\sum_{j=1}^{N} [w_j \prod_{i=1}^{m_j} \exp(-\frac{(x_i - \mu_{ij})^2}{\sigma_{ij}^2})]}{\sum_{j=1}^{N} \prod_{i=1}^{m_j} \exp(-\frac{(x_i - \mu_{ij})^2}{\sigma_{ij}^2})} \tag{11}$$

where N is the number of fuzzy rules (the number of hidden nodes of the RBF network), m_j is the number of premises in the j-th rule (the dimension of the basis function of the j-th hidden node of the RBF network), μ_{ij} and σ_{ij} are parameters of the membership functions (the parameters of the basis functions), and w_j is the consequent parameter of the j-th fuzzy rule (the weight between the j-th hidden node and the output node).

As stated in restriction 6, for an RBF network that is equivalent to a well interpretable fuzzy system, some of its weights should share. However, this constraint is temporarily lifted at this training stage to guarantee that a good solution can be found. Consequently, a conventional learning algorithm based on the gradient method can be directly applied. The algorithm is easy to derive and therefore will not be expounded here.

In addition to parameter adaptation, premises whose membership function is very flat will be set to 'don't care' and rules with a very low activation strength will be deleted. This is effective in simplifying the fuzzy system.

4. Extraction of Fuzzy Rules by Regularization

Since all the parameters have been adapted using the conventional learning algorithm, no weights in the RBF neural network necessarily share the same value when the learning process ends. In this case, the RBF neural network can no longer be converted back to a well interpretable fuzzy system. To extract meaningful fuzzy rules from the trained neural network, we introduce here a novel weight sharing regularization technique. This technique enables the output weights and the parameters of the basis functions of the RBF network to share some certain values so that each fuzzy partitioning has a proper number of fuzzy subsets with well distinguishable membership functions.

4.1. Determination of shared weights

First, it is necessary to find out which weights should share. To this end, we need some tools to measure the similarity of different weights. Euclidean distance, Minkowsky metric and Tanimoto similarity are some available measures. For the sake of simplicity, we use here the Euclidean distance. For any two vectors $V_1 = (v_{11}, v_{12}, \ldots . v_{1m})$ and $V_2 = (v_{21}, v_{22}, \ldots v_{2m})$, their Euclidean distance is expressed as follows:

$$D(V_1, V_2) = \sqrt{\sum_{k=1}^{m} (v_{1k} - v_{2k})^2} \tag{12}$$

In our work, there are two different cases. For the basis functions, each vector has two elements, namely the center and width of the Gaussian functions. For the output weight, there is only one element. We take the basis functions as an example, and if we assume that input x_i has M different basis functions $A_{ij}\,(j = 1, 2, \ldots, M)$ with center μ_{ij} and width σ_{ij}, then specification of shared weights proceeds as follows:

1. Arrange the basis functions A_{ij} in an increasing order according to their center values. Set A_{ij} to U_{ik}, where U_{ik} is a set of weights whose Euclidean distances are within a prescribed threshold of d_i, let $j, k = 1$, $A_i^0 = A_{ij}$.
2. If $D(A_i^0, A_{ij+1}) < d_i$, set A_{ij+1} to U_{ik}; else $k = k + 1$, set A_{ij+1} to U_{ik}, let $A_i^0 = A_{ij+1}$.
3. $j = j + 1$, if $j \leq M$, go to step 2, else stop.

All the basis functions assigned to the same set U_{ik} should share a mutual center and a mutual width value. We find in practice that if we simply select the average center and the average width of a set as the values to be shared, the performance of the extracted fuzzy system will not be satisfactory. Therefore, an adaptive weight sharing regularization technique is proposed in the next subsection.

4.2. Adaptive weight sharing

Reduction of the effective number of parameters in artificial neural networks is believed to be an efficient way to improve their generalization ability. This has resulted in several network regularization techniques, one of which is called weight sharing [7]. The basic idea of weight sharing is to let a single weight be shared among many connections so that the number of adjustable weights in the neural network is limited. If it is not known in advance which weights should share, soft weight sharing [10] can be used.

To extract interpretable fuzzy rules from the trained RBF neural network, an adaptive weight sharing method is proposed here. We do not constrain the shared weights to have the same value, instead, we realize weight sharing by regularizing the RBF neural network, i.e. by adding an extra term to the conventional cost function. Thus, the whole cost function can be defined by:

$$J = E + \lambda \cdot \Omega \tag{13}$$

where E is the usual quadratic error function, λ is the regularization coefficient ($\lambda < 1$) and Ω is a penalty term for weight sharing expressed in the following form:

$$\begin{aligned} \Omega \;=\; & \frac{1}{2}\Big(\sum_i \sum_k \sum_{A_{ij} \in U_{ik}} (\mu_{ij} - \bar{\mu}_{ik})^2 \\ & + \sum_i \sum_k \sum_{A_{ij} \in U_{ik}} (\sigma_{ij} - \bar{\sigma}_{ik})^2\Big) \end{aligned} \tag{14}$$

where $\bar{\mu}_{ik}$ and $\bar{\sigma}_{ik}$ are the center and width values to be shared for the weights in set U_{ik}. The average center and width of each set U_{ik} will be set to their initial values. If Mamdani fuzzy rules are to be extracted, the output weights w_j should also be regularized. In this case, a similar clustering process on the w_j's will be carried out and an extra term should be added to Ω. All the parameters, including the shared weights $\bar{\mu}_{ik}$ and $\bar{\sigma}_{ik}$, will be adjusted by applying the gradient method on the cost function in equation (13):

$$\frac{\partial J}{\partial \mu_{ij}} \Big|_{A_{ij} \in U_{ik}} = \frac{\partial E}{\partial \mu_{ij}} + \lambda(\mu_{ij} - \bar{\mu}_{ik}) \tag{15}$$

$$\frac{\partial J}{\partial \sigma_{ij}} \Big|_{A_{ij}\in U_{ik}} = \frac{\partial E}{\partial \sigma_{ij}} + \lambda(\sigma_{ij} - \bar{\sigma}_{ik}) \qquad (16)$$

$$\frac{\partial J}{\partial \bar{\mu}_{ik}} = -\lambda \sum_{A_{ij}\in U_{ik}} (\mu_{ij} - \bar{\mu}_{ik}) \qquad (17)$$

$$\frac{\partial J}{d a_{ik}} = -\lambda \sum_{A_{ij}\in U_{ik}} (\sigma_{ij} - \bar{\sigma}_{ik}) \qquad (18)$$

5. Simulation Studies

In this section, simulation studies on a prediction problem of the Mackey-Glass time series are carried out to show the feasibility of the proposed method. The Mackey-Glass time series is described by:

$$\dot{x} = \frac{ax(t-\tau)}{1 + x^b(t-\tau)} - cx(t) \qquad (19)$$

where $\tau = 30$, $a = 0.2$, $b = 10$, and $c = 0.1$. One thousand data points are used in the simulation, 500 points for training and the other 500 points for test. Our task is to predict x(t) using $x(t-1), x(t-2)$ and $x(t-3)$. That is to say, the fuzzy system has three inputs and one output. Each input variable is primarily partitioned into 3 fuzzy subsets and the output is partitioned into 4 fuzzy subsets. Therefore, if the fuzzy system takes the standard rule structure, there will be 27 fuzzy rules in total. By using the fuzzy rule generation method with both parameter and structure optimization introduced in section 2, 13 fuzzy rules are generated, which are provided in Table I, where a '*' denotes 'don't care'. It should be pointed out that, according to the consistency definition given in equation (7), a more specified rule will not be considered inconsistent with a more general rule even if their consequents are different (refer to the third and fourth rules in Table I). The mean absolute errors for training and test are both about 0.027.

In order to refine the knowledge obtained by the ES generated fuzzy system, an RBF neural network is constructed according to the fuzzy system. The RBF network is further trained using the 500 training data for 5000 iterations. During training, all the rules whose average activation level is less than 0.01 are deleted. Thus, 6 fuzzy rules remain, which are provided in Table II. The mean absolute errors for training and test are now reduced to about 0.008. From Table II, we notice that for $x(t-2)$, $x(t-1)$ and $x(t)$, no weights share and the distinguishability of different weight values is not good. In other words, the RBF network can not be directly converted back to a fuzzy system that can clearly be interpreted.

The specification process introduced in section 4.1 is applied to the rules in Table II. As a result, $x(t-3)$ has one group, $x(t-2)$ and $x(t)$ have two groups and $x(t-1)$ has three groups. Using the adaptive weight sharing algorithm suggested in the last section, six fuzzy rules are obtained (see Table III), two of which are the same. The mean absolute errors for training and test are both about 0.016.

According to the distribution of the membership functions, a proper linguistic value can be assigned to each fuzzy subset of the inputs (refer to Figure 1, Figure 2 and Figure 3). For example, for $x(t-2)$, "*SMALL*" can be assigned to its membership function $(0.01, 0.7)$ and "*quite SMALL*" can be assigned to the membership function $(0.29, 1.68)$. In this way, the following knowledge concerning the Mackey-Glass time series in terms of well understandable fuzzy rules can be extracted.

- If $x(t-2)$ is *SMALL* and $x(t-1)$ is *BIG*, then $x(t)$ is *BIG*.
- If $x(t-1)$ is *quite SMALL*, then $x(t)$ is *SMALL*.
- If $x(t-2)$ is *quite SMALL* and $x(t-1)$ is *SMALL*, then $x(t)$ is *SMALL*.
- If $x(t-1)$ is *BIG*, then $x(t)$ is *BIG*.

Table 1. The ES Generated Fuzzy System

If			Then
x(t-3)	x(t-2)	x(t-1)	x(t)
*	(0.01,1.35)	([illegible]1,0.66)	1.50
(0.07,0.21)	(0.02,0.34)	*	*0.00*
(0.07,0.21)	*	(0.34,0.69)	*0.00*
(0.07,0.21)	*	*	0.30
(0.07,0.21)	(0. 01,1.35)	*	0.95
*	*	(0.34,0.69)	*0.00*
*	(0.01,1.35)	(0.34,0.69)	*0.00*
(0.07,0.21)	(0.01,1.35)	(1.26,3.60)	*0.00*
*	(0.02,0.34)	*	*0.00*
*	*	(1.21,0.66)	1.50
(0.13,3.97)	(0.01,1.35)	*	1.50
*	(0.25,1.80)	(0.34,0.69)	*0.00*
(0.13,3.97)	(0.02,0.34)	(1.26,3.60)	*0.00*

Table 2. The Fuzzy System After Learning

If			Then
x(t-3)	x(t-2)	x(t-1)	x(t)
*	(0.02,0.84)	(1.04,0.53)	1.50
*	*	(0.18,0.58)	0.03
*	(0.16,1.46)	(0.26,0.12)	0.00
*	*	(1.50,0.22)	1.32
(0.02,3.94)	(0.00,0.38)	*	1.46
*	(0.40,1.89)	(0.11,0.27)	0.0

Table 3. The Extracted Fuzzy System

If			Then
x(t-3)	x(t-2)	x(t- 1)	x(t)
*	(0.01,0.70)	(1.39,0.63)	1.43
*	*	(0.35,0.65)	0.00
*	(0.29,1.68)	(0.25,0.22)	0.00
*	*	(1.39,0.63)	1.43
(0.00,3.93)	(0.01,0.70)	*	1.43
*	(0.29,1.68)	(0.25,0.22)	0.00

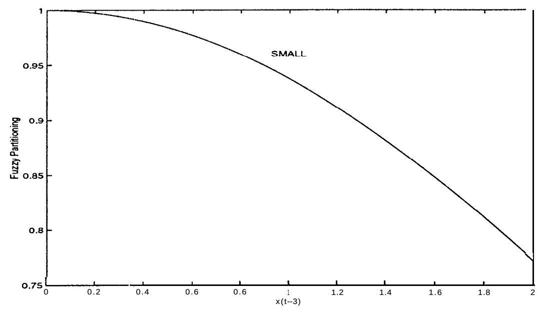

Figure 1. Membership functions for $x(t-3)$.

- If *x(t − 3)* is *SMALL* and ***x(t − 2)*** is *SMALL,* then ***x(t)*** is *BIG.*

It is interesting to notice that the extracted fuzzy rules mainly cover the extreme points of the series. This coincides with the analytical results drawn from some function approximation problems in [6].

6. Conclusion

An approach to rule-based knowledge extraction is described in this paper. Methods for generating complete and consistent fuzzy systems using evolution strategies, fuzzy rule refinement based on RBF neural networks and rule extraction using an adaptive weight sharing algorithm have been presented. .A simple rule extraction study on the Mackey-Glass time series has shown the feasibility of the proposed approach.

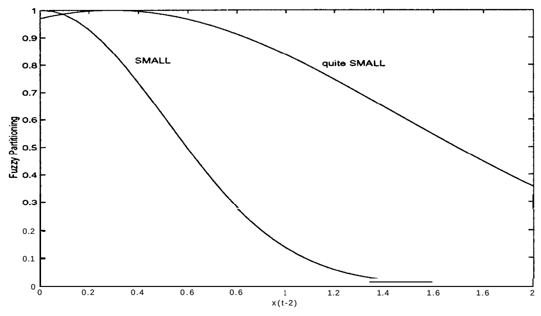

Figure 2. Membership functions for $x(t-2)$.

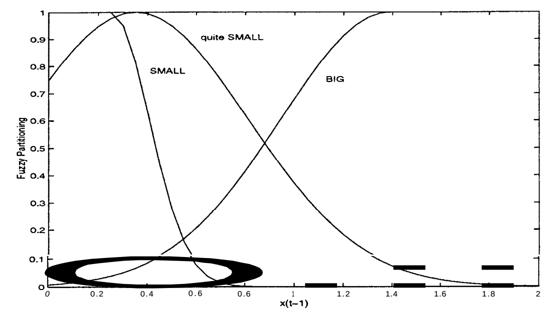

Figure 3. Membership functions for $x(t-1)$.

References

[1] T. Back. Parallel optimization of evolutionary algorithms. In *Parallel Problem Solving From Nature,* pages 418 – 427, 1994.

[2] J. S. R. Jang and C. T. Sun. Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Trans. on Neural Networks,* 4(1): 156 – 158, 1993.

[3] Y. C. Jin. Decentralized adaptive fuzzy control of robot manipulators. *IEEE Trans. on Systems, Man, and Cybernetics,* 28(1), 1998.

[4] Y. C. Jin, W. von Seelen, and B. Sendhoff. Generating fc^3 fuzzy rule systems from data using evolution strategies. *Submitted to IEEE Trans. on Systems, Man, and Cybernetics,* 1997.

[5] C. Karr. Applying genetics to fuzzy logic. *IEEE AI Expert,* 6(2):26 – 33, 1992.

[6] B. Kosko. Optimal fuzzy rules cover extremes. *ht. Journal **of** Intelligent Systems,* 10(2):249 – 255, 1995.

[7] Y. LeCun. Generalization and network design strategies. Technical report, University of Toronto, 1989.

[8] A. Lotfi and A. C. Tsoi. Interpretation preservation of adaptive fuzzy inference systems. *ht. Journal **of** Approaximating Reasoning,* 15(4), 1996.

[9] J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation,* 1:281 -294, 1989.

[10] S. J. Nowlan and G. E. Hinton. Simplifying neural networks by soft weight sharing. *Neural Computation,* 4(4):473 – *493,* 1992.

[11] L. X. Wang and J. Mendel. Generating fuzzy rules by learning from examples. *IEEE Trans. on Systems, Man, and Cybernetics,* 22(6): 1414 – 1427, 1992.

[12] W. Wienholt. *Entwurf neuronaler Netze.* Verlag Harri Deutsch, Thun, 1996.