A staged approach for generation and compression of fuzzy classification rules

Giovanna Castellano and Anna Maria Fanelli
Dipartimento di Informatica
Università di Bari
Via E. Orabona, 4 - 70126 - Bari - ITALY

Abstract - **A staged approach to identify a compact fuzzy classification rule base from numerical data is presented. First, the fuzzy rules are generated by adaptively clustering the input data and defining a relationship between cluster membership values and class labels. Then, the classification accuracy of the resulting fuzzy rules is enhanced by training a neuro-fuzzy network used to model the fuzzy classifier. Finally, the interpretability of the resulting fuzzy classifier is improved via a compression of the fuzzy rule base. Two well known data classification problems are considered to asses the validity of the approach.**

I. Introduction

A growing research topic is the identification of fuzzy rules from the available information about the domain knowledge of a specific task. Such fuzzy rules can be used to build up fuzzy controllers, fuzzy classifiers or to support decision making processes [1], [2], [3]. In many application tasks, fuzzy rules are manually derived from human expert knowledge, but this approach becomes impractical when such a-priori knowledge is not available. Recently several methods have been proposed for automatically generate fuzzy rules from numerical data [4], [5], [6], [7], but few of them can be directly applied to pattern classification problems [5], [8], [9], [10]. The main problem in classification tasks with high-dimensional pattern space is that the number of generated fuzzy rules may become very large. This makes hard a linguistic interpretation of the generated classification rules. Therefore the problem of finding a balance between the rule base size and the accuracy is of considerable importance.

In this paper we propose a staged approach to construct a parsimonious but accurate fuzzy classification rule base from numerical data. In the first stage, fuzzy rules are generated by partitioning the input space into fuzzy regions (clusters), and defining a fuzzy rule for each using the cluster fuzzy membership and the target class labels. In the second stage, to enhance the classification rate of the resulting fuzzy classifier, the generated rules and their parameters are used to initialize the structure and the weights of a neural network which is trained to optimize the rule parameters. In the last stage, the interpretability of the rule base is improved by pruning off unnecessary rules and adjusting the remaining ones so that the classification rate remains unchanged.

Preliminar experimental results over two well-known classification problems are presented.

II. The fuzzy classification scheme

In this section we outline the basics of the adopted fuzzy reasoning scheme for pattern classification problems.

Let us consider a n-dimensional classification problem for which P patterns $\bar{x}^p = (x_1^p, ..., x_n^p)$, $p = 1, 2, ..., P$ are given from m classes $C_1, C_2, ..., C_m$. The task of a pattern classifier is to assign a given pattern $\bar{x}$ to one of the m possible classes based on its features values. Thus, a classification task can be represented as a mapping

$$\psi : X \subset \Re^n \rightarrow \{0,1\}^m$$

where $\psi(\bar{x}) = \bar{c} = (c_1, ..., c_m)$ such that $c_k = 1$ and $c_j = 0$ $(j = 1, ..., m, j \neq k)$.

To solve this classification problem we consider fuzzy rules of the following type:

IF $(x_1^p$ is $A_1^k)$ AND ... AND $(x_n^p$ is $A_n^k)$ THEN $(c_1$ is $b_1^k)$ AND ... AND $(c_m$ is $b_m^k)$

for $k = 1...K$, where K is the number of fuzzy rules, $A_i^k (i = 1...n)$ are fuzzy sets defined over the input variables x_i, and b_j^k are fuzzy singletons defined over the membership value c_j of pattern $\bar{x}^p$ to class $C_j (j = 1, ..., m)$.

Different types of membership functions can be used for the antecedent fuzzy sets. In this paper Gaussian membership functions are used because they are smooth unimodel functions which correspond well with heuristic fuzzy membership function and the parameterized mathematical form aids computation and programming. The type of Gaussian function employed for antecedent fuzzy sets A_i^k, is in the form:

$$\mu_{ik}(x_i) = \exp\left(-\frac{(x_i - w_{ik})^2}{\sigma_{ik}^2}\right) \quad (1)$$

where w_{ik} and σ_{ik} are the center and the width of the gaussian function, respectively.

Once a set of K rules are generated as described in the following section, they are used to classify an unknown pattern $\bar{x}^0 = (x_1^0, ..., x_n^0)$ by an inference mechanism in which Larsen's product operator is used as fuzzy conjunction of fuzzy rules and sum is used as aggregation.

The complete inference results are the class membership values for the pattern $\bar{x}^0$:

$$\hat{c}_j = \frac{\sum_{k=1}^{K} \mu_k(\bar{x}^0) b_j^k}{\sum_{k=1}^{K} \mu_k(\bar{x}^0)} \qquad j = 1, ..., m \tag{2}$$

where

$$\mu_k(\bar{x}^0) = \prod_{i=1}^{n} \mu_{ik}(x_i^0) \quad k = 1, ..., K \tag{3}$$

are the rule activation strengths.

Thus the outputs $\hat{c}_j \in [0, 1]$ of the fuzzy classifier represent the membership degree of the pattern to class C_j. This yields to a "soft" (fuzzy) classification. To obtain hard classification, the highest component of the output vector is mapped to 1 while other components are mapped to 0. In other words, the pattern $\bar{x}^0$ is assigned to the class C_t such that $\hat{c}_t = \max\{\hat{c}_1, \hat{c}_2, ..., \hat{c}_m\}$.

III. Creating rules from data

The generation of fuzzy rules from the available data is made by clustering the input space (i.e. the number of obtained cluster prototypes results independent of the number of classes) and defining the logical relationship between the cluster membership values and the class labels.

A. Clustering the input space

The input space partition is made through an adaptive clustering algorithm, similar to the one developed in [11]. The algorithm divides the input space into hyper-rectangles by applying a series of guillottine cuts. By a guillottine cut, we mean a cut which is made entirely across the subspace to be partitioned: each of the resulting region can then be subjected to independent guillottine cuts. At the beginning of the kth step, the input space is partitioned into k hyper-rectangles. Then, another cut is applied to one of the hyper-rectangles to further partition the entire space into $k + 1$ partitions. At each step, the strategy to decide which dimension to cut and where to cut it, is based on the use of two fuzzy clustering objective functions: a density measure J_D [12] and a typicality measure J_T [3]. They are defined as follows:

$$J_D = \sum_{p=1}^{P} \sum_{s=1}^{P} \{[\sum_{k=1}^{K} (\omega_{pk} - \omega_{sk})^2] - d(\bar{x}^p, \bar{x}^s)^2\}^2$$

$$J_T = \sum_{p=1}^{P} \sum_{k=1}^{K} \omega_{pk} d(\bar{z}_k, \bar{x}^p)^2$$

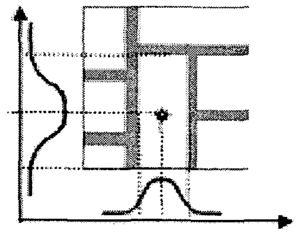

Fig. 1. Input space partition and derivation of membership function parameters

where $\bar{z}_k = (z_{1k}, ..., z_{nk})$ is the center of the kth hyper-rectangle, $d(.,.)$ is the Euclidean distance, and ω_{pk} is the membership value of the pth point $\bar{x}^p$ to the kth hyper-rectangle, which is defined as:

$$\omega_{pk} = \prod_{i=1}^{n} \exp\left(-\frac{(x_i^p - z_{ik})^2}{a_{ik}^2}\right)$$

with parameters a_{ik} and z_{ik} computed depending on the hyper-rectangle resulting from guillotine cuts.

The use of these two objective functions allows to find a meaningful structure for the input fuzzy partition. Density and typicality measures are closely related to the support (i.e. the range of nonzero membership values) and the core (the range of full membership values) of fuzzy sets, respectively. In order to obtain fuzzy sets with strong support (small J_D) and representative core (small J_T) we choose $J = J_D + J_T$ as objective function for the fuzzy partitioning process. At each step, the value of J is computed for the fuzzy partitions resulting from all possible guillottine cuts, and then the partition with the least J value is selected as the next hypothesis to continue the partitioning process.

Indeed, in our context we do not need to find a perfect input space partition, for which the computational cost tends to be high, but just a satisfying partition which would result in a first approximation of the fuzzy rule base to be further enhanced. Accordingly, the partitioning algorithm is stopped as soon as:

$$|J_T(k) - J_T(k-1)| < \epsilon$$

where ϵ is small threshold. In this way, a fuzzy partition of the input space into K clusters is adaptively obtained.

Each cluster center acts as a prototypical data point that represents the antecedent of a fuzzy rule. Calculation of the rule parameters is dependent on the hyper-rectangle defined by a cluster (fig. 1). Precisely, from the components of the cluster center, we derive the centers of the Gaussian membership functions in the antecedents, that is the center w_{ik} of the ith fuzzy set in the kth rule is defined as the ith coordinate z_{ik} of the cluster center. Meanwhile the width of the Gaussian function σ_{ik} is assigned to the value of the cluster radius, i.e. half

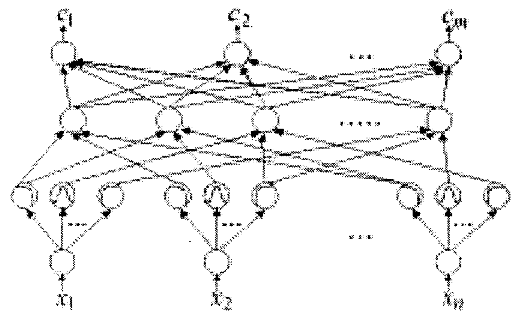

Fig. 2. The neuro-fuzzy network

the length of the hyper-rectangle along the ith component. Thus both the number of rules and their premise parameters come out from the algorithm.

B. Computing the rule consequents

To determine the rule consequent parameters a logical relationship between the structures (clusters) found in the input space and the class labels has to be specified. Specifically, this relation is defined using the information in the cluster membership values ω_{pk}, directly available from the clustering algorithm, and the target vectors $\bar{c}^p = (c_1^p, ..., c_m^p)$, for all $p = 1, ..., P$.

Hence, the consequent values b_j^k of the kth rule are obtained as follows:

$$b_j^k = \frac{\sum_{p=1}^{P} \omega_{pk} c_j^p}{\sum_{p=1}^{P} \omega_{pk}} \qquad j = 1...m \qquad (4)$$

This relation takes into account how much patterns belonging to class C_j are covered by the kth cluster.

IV. Tuning rules

To enhance the generated fuzzy rule base in terms of classification rate, we tune both the antecedent membership functions and the consequent values of each fuzzy rule through a supervised learning stage. To do this, the extracted rules and their parameters are used to determine the structure and to set the initial weights of a 4-layer feedforward neural network (fig. 2). The structure of the network is composed of a set of units $L = L_1 \cup L_2 \cup L_3 \cup L_4$ and a set of connections $U = W \cup V$.

Units of the network have the following specifications:
1. Units $i \in L_1$ simply supply input features $x_i (i = 1, ..., n)$ to units in L_2.
2. Units $i_k \in L_2$ compute membership values $\{\mu_{ik}(x_i)\}$ according to (1).
3. Units $k \in L_3$ compute rule activation strengths $\mu_k(\bar{x})(k = 1, ..., K)$ according to (3).
4. Units $j \in L_4$ compute the class membership values $\hat{c}_j (j = 1, ..., m)$ according to (2).

The sets of connections are defined as follows:

$$W = \{(i_k, k) | i_k \in L_2, k \in L_3\}$$

$$V = \{(k, j) | k \in L_3, j \in L_4\}$$

Each connection $(i_k, k) \in W$ is associated with a pair of weights (w_{ik}, σ_{ik}) corresponding to center and width of the gaussian membership function μ_{ik}. Each connection $(k, j) \in V$ is associated with a weight v_{kj} corresponding to the jth consequent value in the kth rule.

The training of this network performs an optimal adjustment of weights (w_{ik}, σ_{ik}) and v_{kj}. The learning algorithm is a back-propagation-like algorithm based on a gradient-descent technique [13], [14]. Given a training set $T = \{(\bar{x}^p, \bar{c}^p) | \bar{x}^p \in X, \bar{c}^p \in \{0,1\}^m\}_{p=1,...,P}$, the goal is to minimize the error function $E = \frac{1}{P} \sum_{p=1}^{P} E_p$ with $E_p = \frac{1}{2} \sum_{j=1}^{m} (c_j^p - \hat{c}_j^p)^2$ where $\hat{c}_j^p$ is the jth output of the neuro-fuzzy network for pattern $\bar{x}^p$ and c_j^p is the desired class label. The update formula for a generic weight α is $\Delta\alpha = -\eta \frac{\partial E}{\partial \alpha}$ where η is the learning rate.

In summary, the learning algorithm is as follows.

(Rule Tuning Algorithm *)*

1. initialize weights $w_{ij} \in W$ and $\sigma_{ij} \in W$ with center and width of membership functions determined by clustering
2. initialize weights $v_{kj} \in V$ with degree values b_j^k computed in (4)
3. select the next sample $(\bar{x}^p, \bar{c}^p) \in T$, propagate it through the neuro-fuzzy network and determine the output class membership values $[\hat{c}_1^p, ..., \hat{c}_m^p]$
4. compute the error terms for units $j \in L_4$

$$\delta_j^{(4)} = \frac{c_j^p - \hat{c}_j^p}{\sum_{k=1}^{K} \mu_k(\bar{x}^p)}$$

5. update weights $v_{kj} \in V$ by adding the update quantity

$$\Delta v_{kj} = -\eta \frac{\partial E}{\partial v_{kj}} = \eta \delta_j^{(4)} \mu_k(\bar{x}^p)$$

6. compute the error terms for units $k \in L_3$

$$\delta_k^{(3)} = -\sum_{j=1}^{m} \delta_j^{(4)} (v_{kj} - \hat{c}_j^p)$$

7. compute the error terms for units $i_k \in L_2$

$$\delta_{ik}^{(2)} = \sum_{k=1}^{K} \delta_k^{(3)} \mu_k(\bar{x}^p)$$

8. update weights $w_{ik} \in W$ by adding the update quantity

$$\Delta w_{ik} = -\eta \frac{\partial E}{\partial w_{ik}} = \eta \delta_{ik}^{(2)} \frac{2(x_i^p - w_{ik})}{\sigma_{ik}^2}$$

9. update weights $\sigma_{ik} \in W$ by adding the update quantity

$$\Delta\sigma_{ik} = -\eta\frac{\partial E}{\partial\sigma_{ik}} = \eta\delta_{ik}^{(2)}\frac{2(x_i^p - w_{ik})^2}{\sigma_{ik}^3}$$

10. if $E < \epsilon$ then go to step 11. else go to step 3.
11. End

V. Reducing rules

In this last stage, the enhanced fuzzy rule base is simplified to improve its interpretability while preserving the classification accuracy. The rule reduction procedure is an extended version of that we previously developed for simplifying neuro-fuzzy models [15].

The unnecessary rules are iteratively pruned off and the remaining ones are adjusted so that the classification rate remains approximately unchanged. At each step, a rule is identified to be removed and the remaining ones are properly updated. Once a criterion has been defined to choose the rule to be removed, the rule reduction can be stated as follows.

(Rule Reduction Algorithm *)*

1. identify the unit $h \in L_3$ to be removed
2. remove ingoing connections $\{(i_h, h)\}_{i_h \in L_2}$
3. remove outgoing connections $\{(h, j)\}_{j \in L_4}$
4. remove units $i_h \in L_2$
5. remove unit $h \in L_3$
6. update remaining weights $\{v_{kj}\}_{k \in L_3, j \in L_4}$ by adding appropriate adjusting factors δ_{kj} (computed as specified below)
7. if *stopping condition* is met then go to step 8 else go to step 1
8. End

The update quantities δ_{kj}'s are derived by imposing that the net input of each output unit $j \in L_4$ remains approximately unchanged after the elimination of unit $h \in L_3$. This amounts to requiring that, for each training pattern $\bar{x}^p$ and for each unit $j \in L_4$, the following relation holds:

$$\sum_{k \in L_3} v_{kj}\mu_k(\bar{x}^p) = \sum_{k \in L_3 - \{h\}} (v_{kj} + \delta_{kj})\mu_k(\bar{x}^p) \qquad (5)$$

Simple algebraic manipulations yield the following linear system:

$$\sum_{k \in L_3 - \{h\}} \delta_{kj}\mu_k(\bar{x}^p) = v_{hj}\mu_h(\bar{x}^p) \qquad (6)$$

The quantities δ_{kj}'s are then computed by solving the linear system (6) in the least-squares sense throught an efficient preconditioned Conjugate-Gradient method [16]. The criterion for identifying the unit (rule) to be removed at each step has been suggested by the adopted least-squares method. Such a method provides a better solution with faster convergence if the system being solved has a small known term vector $\{v_{hj}\mu_h(\bar{x}^p)\}$ (in terms of Euclidean norm). Since in system (6) the known terms depend essentially on the unit $h \in L_3$ being removed, our idea is to choose the unit for which the norm of the known term vector is minimum. The algorithm is stopped before the performance of the reduced network worsens significantly.

VI. Experimental results

The performance of the proposed approach has been evaluated on two well known classification datasets: the Iris data [17], and the Pima Indians Diabetes data [18].

A. Iris data

The classification problem of the Iris data consists of classifying three species of iris flowers (setosa: C_1, versicolor: C_2 and verginica: C_3). A sample is a four-dimensional pattern vector (x_1, x_2, x_3, x_4) representing four attributes of the iris flower (sepal length, sepal width, petal length, petal width). There are 150 samples for this problem, 50 of each class 3. This data set was divided into a training set and a test set. Both subsets were the same used in [8].

In the first stage of the proposed approach 14 fuzzy rules were extracted from the training set, resulting in a fuzzy classifier with a classification rate of 85.3%. Then, the accuracy of the generated fuzzy classifier was improved to 100% by training a 4-56-14-3 neuro-fuzzy network, whose weights were initialized with the premise and consequence parameters of the identified rules. Subsequently, the enhanced rule base was reduced by simplifying the structure of the network until the classification rate over the test set worsened for more than 1%. Table I summarizes the classification rates of the fuzzy classifier obtained after the three stages of the proposed approach. It can be seen that 9 rules are removed while leaving completely unchanged the classification rate of the fuzzy system. Hence the method provided a fuzzy classifier with 5 rules (the final rules are depicted in fig. 3) and a classification rate of 94.6% on the test set.

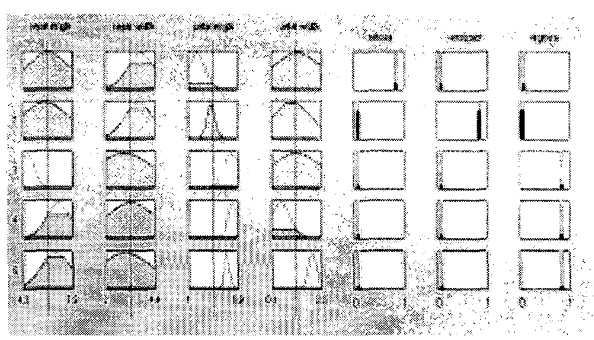

Fig. 3. The final fuzzy rules for the Iris Problem

TABLE I

Results obtained at each stage of the proposed approach for Iris Data

stage	#rules	% classification
rule generation	14	85.3
rule tuning	14	100.0
rule reduction	5	100.0

TABLE II

Comparison among various classifiers for Iris data in terms of generalization

classifier	% classification	# miscl.	# rules
Fuzzy min-max network [8]	97.3	2	48
Adaptive fuzzy classifier in [19]	93.3	5	28
Fuzzy classifier in [20]	97.3	2	17
Neuro-fuzzy classifier in [21]	97.3	2	10
Our fuzzy classifier	94.6	4	5

In Table II the performance of this fuzzy classifier was compared with other fuzzy classifiers on the same test set in terms of number of rules, number of misclassified patterns and classification rate. It can be seen that the fuzzy classifier defined by our method outperforms the other ones developed in literature in terms of simplicity, providing the smaller number of rules.

Even though the method proposed in [5] generates a fuzzy classifier with a better accuracy (classification rate of 97.3%), the structure of the rule base is more complex (i.e. 17 rules). Besides, the same authors obtain a number of misclassified test data equal to 6 (i.e. 92.0% classification rate) with the same number of rules as our final classifier (i.e. 5 rules).

B. Pima Indians Diabetes

The data considered in this subsection were collected by US National Institute of Diabetes and Kidney Diseases [18]. A population of women of Pima Indian heritage who were at least 21 years old was tested for diabetes. For each woman the following variables were collected:

- number of pregnancies
- plasma glucose concentrations in an oral glucose tolerance test
- diastolic blood pressure (mmHg)
- triceps skin fold thickness (mm)
- body mass index (weight in kg/(height in m)^2)
- diabetes pedigree function
- age in years

The training set and the test set consist of 200 patterns and 332 patterns, respectively. Both training and test set were taken from [18] who reports that the best classification methods for this problem provide about a 20% of false classification on the test set.

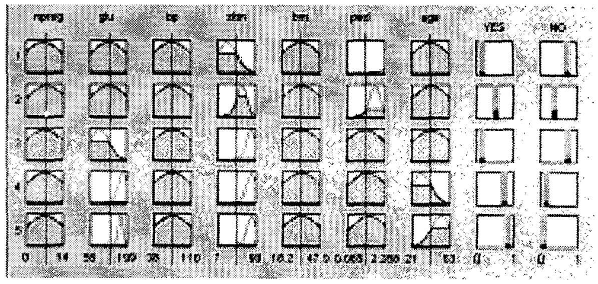

Fig. 4. Final rule base for the Pima Indians Diabetes Problem

TABLE III

Results obtained at each stage of the proposed approach for Pima Indians data

stage	#rules	% classification	
		Train set	Test set
rule generation	13	66.5	67.2
rule tuning	13	72.5	76.0
rule reduction	5	71.5	76.0

For this classification problem, 13 fuzzy rules were extracted after the first stage resulting in a fuzzy classifier with a classification rate of 66.5% on the training set. Then, the fuzzy classification rules were tuned to improve accuracy by training an appropriate neuro-fuzzy network initialized with the parameters of the 13 extracted rules. After training, the classification rate improved to 72.5% on the training set, resulting in a classification rate of 76.0% over the test set. Finally, the rule reduction algorithm (stopped, again, when the classification rate over the test set worsened for more than 1%) reduced the number of rules from 13 to 5 while leaving completely unchanged the classification rate of the fuzzy system on the test set (see Table III).

The same results in terms of number of rules and classification rate on the test set was obtained in [22]. However, it is worthwile noting that our approach provides this results in a completely automatic fashion. Conversely, in [22] the number of rules is not automatically generated but several neurofuzzy networks with different number of rules must be trained in order to evaluate the best final classifier.

VII. Conclusions

In this paper a staged approach to automatically identify a fuzzy classification rule base from numerical data has been described. Such an approach provides good solutions in terms of balance between rule base size and classification accuracy. Promising results on two well-known classification tasks have been obtained. More extensive tests are in progress aiming at evaluating the performance of the approach to high dimensional classification problems.

References

[1] C.C. Lee. Fuzzy logic in control systems: Fuzzy logic controller - part 1 and part 2. *IEEE Transactions on Systems, Man and Cybernetics*, 20(2):404–435, Mar 1990.

[2] D.G. Schwartz, G.J. Klir, H.W. Lewis III, and Y. Ezawa. Applications of fuzzy sets and approximate reasoning. *Proceedings of the IEEE*, 82(4):482–497, April 1994.

[3] J. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum, New York, 1981.

[4] L. X. Wang and J. M. Mendel. Generating fuzzy rules by learning from example. *IEEE Trans. Sys., Man, Cybern.*, 22(6):1414–1427, 1992.

[5] S. Abe and M. Lan. Fuzzy rules extraction directly from numerical data for function approximation. *IEEE Transactions on Systems, Man and Cybernetics*, 25(1):119–129, January 1995.

[6] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modelling and control. *IEEE Transactions on Systems, Man and Cybernetics*, 15(1):116–132, January 1985.

[7] M. Delgado, A.F. Gomez-Skorneta, and F. Martin. A fuzzy clustering-based rapid prototyping for fuzzy rule-based modelling. *IEEE Transactions on Fuzzy Systems*, 5(2), May 1997.

[8] P.K. Simpson. Fuzzy min-max neural networks - part 1: Classification. *IEEE Transactions on Neural Networks*, 3(5):776–786, 1992.

[9] S.K. Pal and S. Mitra. Multilayer perceptron, fuzzy sets, and classification. *IEEE Transactions on Neural Networks*, 3:683–697, Sept 1992.

[10] W. Pedrycz. Fuzzy sets in pattern recognition: Methodology and methods. *Pattern Recognition*, 23(1):121–146, 1990.

[11] C.T. Sun. Rule-base structure identification in an adaptive-network-based fuzzy inference system. *IEEE Trans. on Fuzzy Systems*, 2(1):64–73, 1994.

[12] E.H. Ruspini. Numerical methods for fuzzy clustering. *Inform. Sci.*, 2:319–350, 1970.

[13] J.S.R. Jang and C.T. Sun. Neuro-fuzzy modeling and control. *Proc. of the IEEE*, 83(3):378–406, March 1995.

[14] C-T. Lin and C.S.G. Lee. Neural-network-based fuzzy logic control and decision system. *IEEE Transactions on Computers*, 40(12):1320–1336, December 1991.

[15] G. Castellano and A.M. Fanelli. Simplifying a neuro-fuzzy model. *Neural Processing Letters*, 4(2):75–81, November 1996.

[16] A. Bjiörck and T.Elfving. Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations. *BIT*, 19:145–163, 1979.

[17] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Ann. Eugen.*, 7:179–188, 1936.

[18] B.R. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, 1996.

[19] K. Nozaki, H. Ishibuchi, and H. Tanaka. Adaptive fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems*, 4(3):238–250, August 1996.

[20] S. Abe and M. Lan. A method for fuzzy rules extraction directly from numerical data and its application for function approximation. *IEEE Transactions on Systems, Man and Cybernetics*, 25(1):119–129, January 1995.

[21] D. Nauck and R. Kruse. Nefclass - a neuro-fuzzy approach for the classification of data. In *Proceedings of the 1995 ACM Symposium on Applied Computing*, Nashville, Feb 1995.

[22] J. Leski and E.Czogala. A new artificial neural network based fuzzy inference system with moving consequents in if-then rules and selected applications. *Fuzzy Sets and Systems*, 108:289–297, 1999.