Printed on the Proceedings of the Third
IEEE International Conference
on Fuzzy Systems, IEEE-FUZZ94
Orlando, Florida
pp. 808-812, IEEE 1994

Semantic Phase Transition in a Classifier Based on an Adaptive Fuzzy System

F. Casalino, F. Masulli, A. Sperduti and F. Vannucci

***Abstract*— In this paper we investigate how many rules a particular adaptive fuzzy system, very similar to a feed-forward neural network, needs to perform digits classification. Both training of systems with different numbers of fuzzy rules and the application of a pruning technique to a non minimal system support the statement that one rule for each class is needed. In particular, a semantic phase transition is observed when a new rule is added to a system with 9 rules. This behaviour, which is not common for a feed-forward neural network, could be ascribed to the derivation of the studied system from a fuzzy logic framework.**

I. Introduction

Fuzzy systems have been successfully applied in the field of process control [1]. The strength of these systems is given by the possibility to naturally integrate linguistic rules describing the process into the system. In recent years, adaptive methods which can integrate also knowledge expressed in numeric form have been proposed as well. These systems (e.g., ANFIS [2], Mendel's system [3]) work either by modifying the system parameters or by dynamically adding new rules to it. We applied Adative Fuzzy Systems [4] to handwritten digits classification obtaining good results, especially regarding the speed of training [5]. In this paper we investigate experimentally how many rules the fuzzy system needs to work properly in classification task. We have explored two different approaches: the first one consisted in training networks with an increasing number of rules and using a large training set. The second approach (already proposed in the neural network literature [6, 7]) was based on a *pruning* technique and was tested on a small training set. Our results show that at least ten rules, one for each class, are necessary to perform the handwritten digit classification task. Moreover, a semantic phase transition is observed when a new rule is added to a system with nine rules: nine rules are not able to capture the semantic structure of the training set since the digits of one class are misclassified, while ten rules are sufficient to give a good approximation of such structure.

The paper is organized as follows. The Adaptive Fuzzy System is presented in Section II, data set and preprocessing in Section III. Our approach to study how many rules are needed by the system is outlined in Section IV and the pruning technique discussed in Section V. Results and discussion are presented in Section VI.

II. The Adaptive Fuzzy System

An Adaptive Fuzzy System (AFS) [4] is a feed-forward network that can be seen as a Multi-Layer Perceptron with just one hidden layer whose units correspond to the fuzzy rules. Specifically, if there are K units in the input layer, J rules in the hidden layer and I units in the output layer, the activation of those rules can be written as:

$$R_j = \prod_k \mu_{jk}(x_k), \qquad (1)$$

where $k \in [1, K]$, $j \in [1, J]$, $i \in [1, I]$, and the quantity $\mu_{jk}(x_k)$ represents the value of the membership function of the component x_k of the input vector for the j-th rule.

We defined the membership function as:

$$\mu_{jk}(x_k) = \exp(-\frac{|x_k - m_{jk}|^2}{2\sigma_{jk}^2}), \qquad (2)$$

so that the receptive fields mutually overlap.

This work was supported by grants from CNR-Progetto Strategico Reti Neurali, GNCB-CNR, Consorzio INFM and MURST. Part of this work was carried out in Summer 1993 while F. Masulli was a Senior Visiting Scientist at the International Computer Science Institute in Berkeley, USA. We thank Maurizio Martelli for helpful discussions.

F. Casalino is with the DISI - Department of Computer and Information Sciences , University of Genoa, Viale Benedetto XV, 3 - 16132 Genoa, Italy. His E-mail is casalino@genova.infn.it.

F. Masulli is with the Department of Physics, University of Genoa, Via Dodecaneso 33 - 16146 Genoa, Italy. His E-mail is masulli@genova.infn.it.

A. Sperduti is with the Department of Computer Science, University of Pisa, Corso Italia, 40 - 56125 Pisa, Italy. His E-mail is perso@di.unipi.it.

F. Vannucci is with the INFM Research Unit of Genoa, Via Dodecaneso 33 - 16146 Genoa, Italy.

The values of the output units are obtained using a defuzzification rule based on the centroid [8, 9]:

$$y_i = \frac{\sum_j R_j s_{ij}}{\sum_j R_j}. \quad (3)$$

The AFS is trained to work as a classifier by minimizing the Mean Square Error between the output of the net and the label vector $\vec{l}$, whose components are defined as follows:

$$l_j = \begin{cases} 1 & \text{if the example belongs to the } i\text{-th class,} \\ 0 & \text{otherwise.} \end{cases}$$

The learning formulas for the the system's parameters (i.e., m_{jk}, σ_{jk} and s_{ij}) are obtained using the Back-Propagation technique (see [4]):

$$\Delta s_{ij} = \eta_s [t_i - y_i] \psi_j \quad (4)$$

$$\Delta m_{jk} = \eta_m \sum_i [t_i - y_i][s_{ij} - y_i]\psi_j [x_k - m_{jk}]/\sigma_{jk}^2 \quad (5)$$

$$\Delta \sigma_{jk} = \eta_\sigma \sum_i [t_i - y_i][s_{ij} - y_i]\psi_j [x_k - m_{jk}]^2/\sigma_{jk}^3 \quad (6)$$

where

$$\psi_j = \frac{\prod_k \mu_{jk}(x_k)}{\sum_j \prod_k \mu_{jk}(x_k)} \quad (7)$$

represents the normalized activation of rule j, and η_s, η_m, η_σ the learning rates of s_{ij}, m_{jk}, σ_{jk}. In our implementation, we have adaptive learning rates, which improve considerably the convergence speed [10]. It is worth to notice that the approach to the classification using fuzzy systems could allow us to insert the available a-priori knowledge into the rules before the training phase, and to pull out the learned rules after the training.

III. Data Set and Preprocessing

The examples of the training set and the test set were extracted from the NIST-3 CD-ROM [11] that contains 313389 segmented characters in 128×128 binary image format and their corresponding labels in ASCII format.

During the preprocessing phase, the current character is normalized to a 32×32 format. A low-pass filter is applied in order to fill small holes and remove some small noisy spots, next the character undergoes a shear transform. Finally, the dimensions of the input space are further reduced, to represent each character as a 64-component vector. Each component is associated with the number of black pixels in disjoint 4×4 image subsquares.

The resulting data-base contains a learning set and a test set, both of $10,000$ decimal numerals.

IV. How Many Rules?

In order to assess how many rules are needed by our system to reach a reasonable level of classification, we have explored two different approaches. The first one consists in training networks with a different number of rules. The aim is to show that the network needs at least one rule for each class. This statement is supported by the fact that our system is a simplified inferential fuzzy system for which it is explicitly required each rule to specialize on one class.

The second approach, of which we report preliminary results, starts from a network with more that 10 rules and then uses a *pruning* technique to reduce the number of rules. The pruning technique we have used has been already proposed in the neural network literature [6, 7]. In the following we describe it.

V. The Pruning Technique

The pruning technique we have used is based on linear redundancy removal in the activation space. Linear redundancy in the activation space of the units occurs when the activation of one unit can be expressed as linear combination of the activation of some of the remaining units, over the training set. In a feed-forward neural network, when linear redundancy is present, the redundant unit(s) can be removed by transforming consistently the set of weights of the network. The transformation of the set of weights is defined according to the coefficients of the linear combination expressing the linear redundancy.

Linear redundancy in the activation space can be discovered via classical methods by computational algebra (see for example [12]), however these methods, in general, involve global computations. In the following we show how a neural network with a single layer of linear units can perform the task using only local computations.

A. The Network

In this subsection we present a single layer network able to discover linear redundancy into the activation space of the system's rules. The problem is to test if a vector V is linearly dependent on a set of k given vectors V_i. The basic idea is to transform this problem into a learning problem: the set of vectors V_i are joined together to form a training set and the components of the vector V used as targets for the patterns in the training set. Specifically, the j-th training pattern consists of the j-th component of each vector V_i and the associated desired output is the j-th component of the V vector. For learning we have to use a single linear node with k incoming

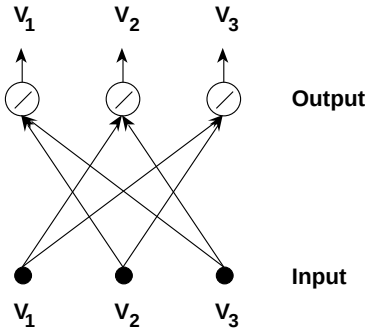


Figure 1: The pruning network.

connections and fixed null bias. Notice that, if one of the vectors is composed of all ones it can be removed by allowing any value for the bias. The delta rule with standard mean square error E is employed to adjust the weights w_i, initially set to zero. If V is linearly dependent on the other vectors, then learning converges and E is set to zero, otherwise learning does not converge and E is set to a non zero value. In the first case, the vector V can be expressed as:

$$V = \sum_{i=1}^{k} w_i V_i. \tag{8}$$

In the second case, V is linearly independent with respect to the vectors V_i and the value of E gives a measure of the similarity between vector V and vector $V' = \sum_{i=1}^{k} w_i V_i$.

In general, we are given a set of vectors (activations of the rules) and we have to discover which vector (rule) can be best approximated by the others on the training set. This corresponds to test for each vector how much the remaining vectors are able to approximate it. Since each test can be performed independently, we can use a single layer network with linear units to execute in parallel the tests (see Figure 1). The network if fully connected except that the i-th input is not connected to the i-th unit. The network is forced to learn the identity function. In this way, at the end of training, each vector is approximated by a linear combination of the other vectors.

B. Application to the Adaptive Fuzzy System

In the neural network framework, the above network is able to perform a *constructive pruning* since it is able both to discover relationships among units at different levels in the network and to use the coefficients of the linear combination to eventually construct new connections after the pruning. In the framework of adaptive fuzzy systems, the network can be exploited in different ways. First of all, suppose that for some index h we have:

$$R_h = \sum_{i=1, i\neq h}^{k} \alpha_i R_i, \tag{9}$$

then the defuzzification rule can be rewritten as:

$$y_i = \frac{\sum_{j,j\neq h} R_j s_{ij} + R_h s_{ih}}{\sum_j R_j} = \tag{10}$$

$$= \frac{\sum_{j,j\neq h} R_j (s_{ij} + \alpha_i s_{ih})}{\sum_j R_j} =$$

$$= \gamma_h \frac{\sum_{j,j\neq h} R_j (s_{ij} + \alpha_i s_{ih})}{\sum_{j,j\neq h} R_j} = \gamma_h \hat{y}_i,$$

where

$$\gamma_h = \frac{\sum_{j,j\neq h} R_j}{\sum_j R_j}. \tag{11}$$

Thus the output $\hat{y}_i$ of the system obtained by removing rule h and transforming the weights s_{ij} according to the coefficients α_i, is equivalent to the original output y_i rescaled by a factor $\frac{1}{\gamma_h}$ which depends on the particular pattern in input. Since the relationship among the output units do not change, we obtain a classifier with the same discriminant ability but with $k-1$ rules.

In general, it will be difficult to discover an *exact* linear redundancy in the system, however approximations of it are frequent both in neural networks and AFSs. Because of these approximations, the transformation of the weights according to the coefficients of the linear combination will be approximated as well. Consequently, after the removal of one rule, some epochs of learning may be necessary to recover the original performances.

A second way to exploit the network, suggested by the above consideration, is to run the network only for a certain amount of epochs, e.g., twelve epochs, and then remove the rule with the minimum value of E, without computing new weights. Learning is then performed only if needed. The undelying idea is that we do not want to spend time computing an accurate approximation of the coefficients of the linear combination when these need also to be changed by learning. Thus a few epochs by the pruning network are sufficient to discover the rule of the AFS that must be removed. In this paper we give some preliminary results using this approach.

VI. Results and discussion

A. First Method: Three Different AFS Configurations

We analyze the results obtained using three differents AFS configurations. We trained three AFSs

Training set		AFS_8	AFS_9	AFS_{10}
Class	# pattern	%	%	%
0	1019	3.43	4.12	4.51
1	1127	2.66	2.13	2.30
2	982	97.04	95.60	12.62
3	1049	7.34	8.30	8.57
4	944	6.03	7.41	8.15
5	870	100.0	17.01	18.85
6	980	4.08	6.32	6.93
7	1035	5.12	4.83	6.08
8	1011	12.75	13.64	10.68
9	983	4.17	4.27	4.57

Table 1: Training set error rates on 10,000 patterns by three different AFS (8, 9, and 10 rules).

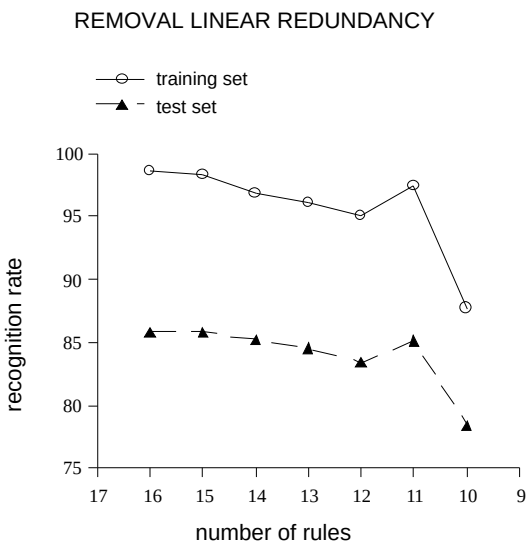


Figure 2: Test and training success rates obtained by pruning the rules in an AFS (1,000 digits).

with 8, 9 and 10 fuzzy rules, respectively, on a training set of 10,000 digits.

Table 1 gives, for each class of digits, the number of patterns in the training set, and the error rates on the training set for the AFSs with 8, 9, and 10 rules. We point out that, if the rules are less than 10, one class is not recognized at all in the case AFS_9, and two classes for AFS_8, while the other classes are well recognized. It is worth noting that, in the case of AFS_{10} the error rate is uniformly distributed along all the classes. These facts suggest that, during the training fase, each rule specializes to recognize one of the ten classes of digits.

B. Second Method: Pruning

We applied the pruning network presented in Section V on a reduced training set of 1,000 digits. We started from a trained network with 16 rules and applied the pruning network to it. The second way to exploit the network was used. After each rule removal three epochs of relearning were applied when necessary. Ten relearning epochs were performed after the removal of the 12-th rule. The obtained results on the error rates for the training set and a test set of other 1,000 digits are shown in Figure 2. It can be noted that after the removal of the 11-th rule the performances of the AFS decrease considerably both on the training and test set. Removing the 10-th rule results into a drop of the AFS's performances. These results are consistent with the ones obtained by the first method. They also show that the pruning technique is able to reach a good approximation of the minimal number of rules needed in the classification task. However, more data, also in different classification tasks, are needed to confirm the performances of the pruning network.

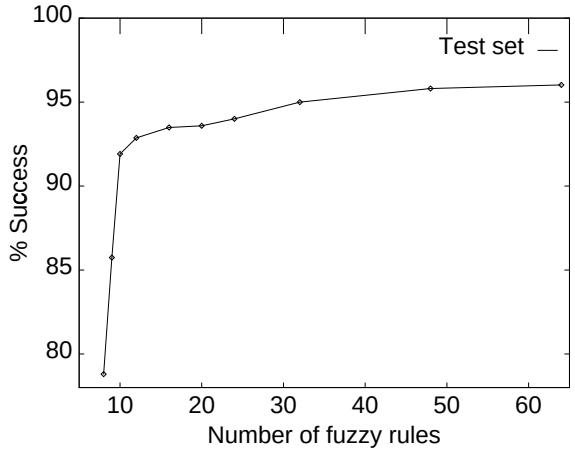
C. Semantic Phase Transition

The results presented above shown a semantic phase transition among 9 and 10 rules: 9 rules are not able to capture the semantic structure of the training set, 10 rules do. Specifically, Figure 2 shows the test training success rates versus the number of fuzzy rules. Adaptive Fuzzy Systems with less of 9 rules show low quality classification performances, but they improve of a factor of about 10% for each new rule. This depends on the lack of at least one rule for each class to recognize. On the contrary, for more than 10 rules the improvements are small, and they correspond to the creation of new rules, specialized to the recognition of different kinds of handwriting styles, occurring for some class of digits in the training set.

It is worth noting that, this abrupt change in the behaviour of inferential fuzzy systems in classification tasks, could be ascribed to a semantic "phase transition" from a state of inability to recognize one or more classes (because of the ignorance of the related rules) to a state of sufficient knowledge.

The abrupt change could also be ascribed to the inability of the Back-Propagation algorithm to find a global minimum with the minimal number of rules, however, both the results obtained with the pruning technique and the particular distribution of the residual errors obtained with 8 and 9 rules are against this hypothesis.

In conclusion, it seems reasonable to state that the studied AFS shows a peculiar difference with respect to feed-forward networks (semantic phase transition) which can be due to the derivation of



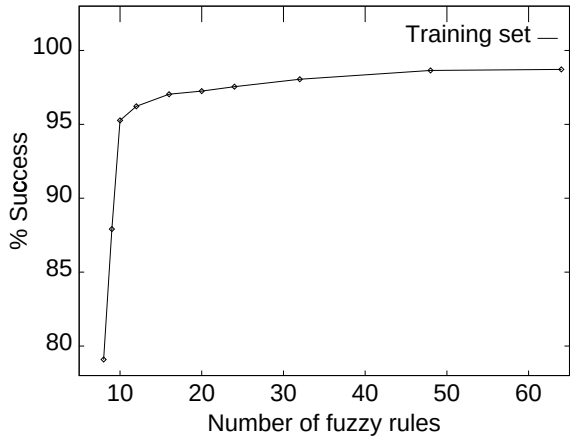


Figure 3: Test and training success rates versus number of fuzzy rules.

the system from a fuzzy logic framework.

References

[1] C. C. Lee, "Fuzzy logic in control systems: fuzzy logic controller-part 1 and 2.", *IEEE Trans. on Systems, Man, and Cybernetics*, vol.20, pp. 404-435, 1990.

[2] J. S. R. Jang, "ANFIS: Adaptive-Network-Based Fuzzy Inference System", *IEEE Trans. on Systems, Man, and Cybernetics*, vol 23, n. 3, pp. 665-684, 1993.

[3] L. Wang, J. M. Mendel "Generating Fuzzy Rules by Learning from Examples", *IEEE Trans. on Systems, Man, and Cybernetics*, vol.22, pp. 1414-1427, 1992.

[4] C. C. Jou, "Comparing Learning Performance of Neural Networks and Fuzzy Systems", *IEEE Int. Conf. on Fuzzy Systems*, pp. 1028-1033, 1993.

[5] F. Masulli, F. Vannucci, F. Casalino, M. Garcia, F. Pasini, M. Penna, "Neural Methods for Handwritten Characters Recognition", *Proceedings of the Workshop on Image Processing by Neural Networks*, December 16th, Rome, Italy, 1993 (in press).

[6] A. Sperduti and A. Starita. "Reducing Linear Redundancy in Neural Networks", In *International Joint Conference on Neural Networks*, pages 612–617, Baltimore, 1992.

[7] A. Sperduti and A. Starita. "A Neural Optimizer Exploiting Linear Redundancy", In *International Joint Conference on Neural Networks*, pages 500–505, Beijing, 1992.

[8] B. Kosko, "Neural Networks and Fuzzy Systems", Prentice Hall, Englewood Cliffs, pp. 317-337, 1992.

[9] C. C. Jou, "On the mapping capabilities of fuzzy inference systems", *Int. Joint Conf. on Neural Networks* ,vol. 2, pp. 708-713, 1992.

[10] F. Casalino, "Fuzzy systems for handwriting recognition", Laurea Thesis in Computer Science, University of Genoa, (in Italian) Genoa, Italy, 1993.

[11] M. D. Garris, R. A. Wilkinson, "NIST Special Database3 Handwritten Segmented Characters", National Institute of Standard and Technology, Gaithesburg, MD, USA, 1992.

[12] S. Y. Kung and Y. H. Hu. "A Frobenius Approximation Reduction Method (FARM) for Determining the Optimal Number of Hidden Units", In *International Joint Conference on Neural Networks*, pages 163–167, Seattle, 1991.