# Learning Techniques for Supervised Fuzzy Classifiers

Francesco Masulli(1,2) and Alessandro Sperduti(3)

(1) Istituto Nazionale per la Fisica della Materia
Via Dodecaneso 33, 16146 Genova, Italy
(2) DISI - Dipartimento di Informatica e
Scienze dell'Informazione - Università di Genova
Via Dodecaneso 35, 16146 Genova, Italy
(3) Dipartimento di Informatica, Università di Pisa, 56125 Pisa, Italy
e-mails: masulli@ge.INFM.it, perso@di.unipi.it

**Abstract**

In this chapter we present a family of learning machines for supervised classification, developed starting from neuro-fuzzy systems based on Fuzzy Basis Functions Networks (FBFNs). FBFNs hold universal function approximation capabilities, can learn their parameters from data, and are able to approximate the Bayes Optimal Classifier in supervised classification tasks. A gradient descent approach for learning is described and the classification capabilities of the trained system are compared with respect to Multi-Layer Perceptrons and Radial Basis Functions networks on a handwritten digit recognition task. Moreover, a simplified version of the model is described that is faster in learning. This simplified version cannot reach the same level of performance as the original model, however, an extended version of it, based on competitive learning, show a significant increasing in classification performance, still retaining a relatively short training time. Further improvement in performance can be obtained by using the k-Nearest-Neighbour Rule for input instances which cannot be classified with high confidence by the system. The increase in performance is paid with a slowdown in the response time. This slowdown, however, can be modulated through the use of *off-line* or *on-line* editing strategies for the k-NN Rule.

## 1 Introduction

The development of Fuzzy Logic Systems (FLSs) [19, 16] requires the definition and tuning of the shapes and sizes of the membership functions. Neuro-fuzzy systems allow the automatic adjustment of these parameters on the basis of training data [28, 13, 12].

In this chapter, we present a FLS with *singleton* fuzzification, *max-product* composition, *product inference* and *height* defuzzification which can be described as a Fuzzy Basis Functions Network (FBFN)[15, 29, 28, 30]. This system is demonstrated to possess universal approximation capabilities and is particularly suited to be trained on data by a gradient descent technique. Moreover, the system is able to approximate a Bayes Optimal Classifier, when trained using a suitable error function.

We show the details on how gradient descent can be applied to the FBF network and we compare, on a handwritten digit recognition task, its performances versus two other learning machines, namely Multi-Layer Perceptrons (MLPs), and Radial Basis Functions Networks (RBF). Moreover, we discuss a structural feature of the proposed model, i.e., the Semantic Phase Transition observed when passing from the training of a model with less rules than classification classes to a model with a number of rules which is equal to or greater than the number of classification classes.

The performance of the FBFNs are very interesting, however, very often, the resulting systems are very complex and may require long training times. Thus, it is of paramount importance to devise simple systems which are fast to train and still hold a very good generalization performance. For this reason, starting from the original formulation, it is possible to derive a simplified version of the model (SFBF), faster in training and still retaining much of the performance in classification of the original model. In fact, at the expenses of an increased training time, but still not so heavy as in the case of the original model, the SFBF model can be improved by introducing competition among rules belonging to the same class.

Finally, we show that performance can be further improved in a significant way by using the k-Nearest-Neighbour (k-NN) Rule [8] over input instances which cannot be classified with high confidence by the SFBF system. This time, the computational cost which must be paid for the increase in generalization is in terms of the average response time of the system. The tradeoff between the increase in performance and response time can be controlled by using both *off-line* or *on-line* editing strategies for the k-Nearest-Neighbour Rule.

## 2 The Fuzzy Basis Function network

Fuzzy Logic Systems with *singleton* fuzzification, *max-product* composition, *product inference* and *height* defuzzification can be represented as [19]

$$y = f(\mathbf{x}) = \sum_{l=1}^{M} \overline{y}^{\,l} \phi_l(\mathbf{x}) \tag{1}$$

where $\overline{y}^{\,l}$ denote the center of gravity of the output fuzzy set, and $\phi_l(\mathbf{x})$ are called *fuzzy basis functions* and are given by

$$\phi_l(\mathbf{x}) = \frac{\prod_{i=1}^{p} \mu_{F_i^l}(x_i)}{\sum_{l=1}^{M} \prod_{i=1}^{p} \mu_{F_i^l}(x_i)} \tag{2}$$

where $l = 1, 2, ..., M$. We can refer to those FLS as *fuzzy basis expansions* or *networks of fuzzy basis functions* (FBF network)[1]. The relationships between fuzzy basis expansions and other basis functions have been extensively studied in [15].

It is worth noting that the FLS with universal function property studied by Mendel and Wang [29, 28], which is a singleton FLS using product inference, product implication, Gaussian membership and height defuzzification, can be rewritten as a FBF network expansion[2]. The universal function approximation property gives a strong mathematical ground when applying FLSs in critical applications, ranging from control, to time series prediction, to pattern recognition.

In this chapter we study a fuzzy logic system based on a Multi-Input-Multi-Output (MIMO) version of this FBF network. Specifically, if there are $K$ units in the input layer, $J$ fuzzy inference rules and $I$ outputs, the rule activations can be expressed as:

$$r_j \;=\; \prod_k \mu_{jk}(x_k), \tag{3}$$

where $\mu_{jk}(x_k)$ is the value of the membership function of the component $x_k$ of the input vector for the $j$th rule and is defined as:

$$\mu_{jk}(x_k) = \exp\left(-\frac{(x_k - m_{jk})^2}{2\sigma_{jk}^2}\right), \tag{4}$$

where $m_{jk}$ and $\sigma_{jk}^2$ are the means and the variances of the Gaussian membership functions. The values of the output units are:

$$y_i = \frac{\sum_j r_j \overline{y}_{ij}}{\sum_j r_j} = \sum_j \overline{y}_{ij} \phi_j(\mathbf{x}) \;, \tag{5}$$

where $\overline{y}_{ij}$ is the center of gravity of the output fuzzy membership function of the $j$th rule associated with the output $y_i$,[3] and

$$\phi_j = \frac{\prod_k \mu_{jk}(x_k)}{\sum_j \prod_k \mu_{jk}(x_k)} \tag{6}$$

is the fuzzy basis function associated to rule $j$, and represents its normalized activation.

[1] In [19] fuzzy basis expansions for FLS with nonsingleton fuzzification are also introduced.

[2] Mouzouris and Mendel [21] do the same for a nonsingleton FLS that uses a range of t-norms, arbitrary membership functions and modified height defuzzification.

[3] Without loss of generality, we could assume that the fuzzy membership functions are singletons ($\overline{y}_{ij} \equiv s_{ij}$).

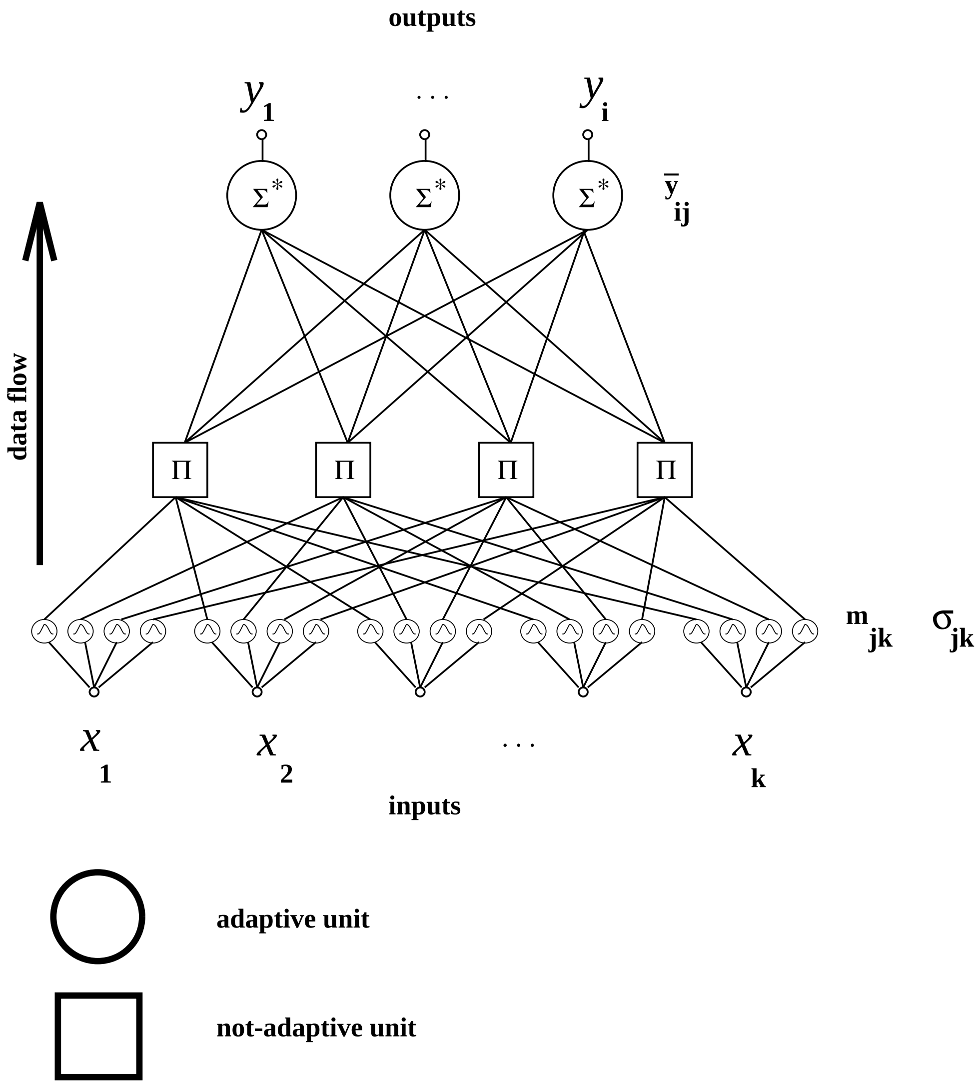



Figure 1: Connectionist interpretation of the FBF network.

The FBF network can be regarded as a feedforward connectionist system with one hidden layer whose units correspond to the fuzzy rules. In Figure 1 a connectionist interpretation of the FBF network is shown. In the drawing, circles represent adaptive units, while squares stand for not-adaptive units, and free parameters are evidenced near the corresponding units.

The FBF network can be identified both by exploiting the linguistic knowledge available (*structure identification problem*) [16] and by using the information contained in a data set (*parameter estimation problem*) [16].

## 3 Bayes Optimal Classifier Approximation

In the neural network literature, it has been demonstrated that a classifier based on an MLP can approximate the Bayes optimal discriminant function, for suitable choices of the cost function to be minimized during the training phase, and in the large training set limit [25, 11, 14, 20, 2].

In [18], we generalized this result and demonstrated, following Ruck et al. [25], that *a classifier based on a system holding the universal function approximation property can approximate the Bayes discriminant function*[4]. The demonstration we present here concerns the two-class case.

Let $c_1$ e $c_2$ be two classes of patterns. Let $\chi_i$ be the set of all possible patterns $\mathbf{x}$ belonging to class $c_i$;

[4]As pointed out by Ruck et al. [25], this demonstration is not based on any assumption about the particular feedforward system used; only the system's capability for function approximation is assumed.

then $\chi = \chi_1 \cup \chi_2$ represents the set of all patterns. The training set consists of a subset of possible patterns belonging to the two classes. In general, it is a finite set $X = X_1 \cup X_2$, where $X_1 = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{n_1}\} \subset \chi_1$, $X_2 = \{\mathbf{x}_{n_1+1}, ..., \mathbf{x}_{n_1+n_2}\} \subset \chi_2$, and $n_1 + n_2 = N$ .

Let $\mathbf{w}$ be the set of adaptive parameters, $g(\mathbf{x})$ be the discriminant function of a Bayes dichotimizer

$$g(\mathbf{x}) = P(c_1 \mid \mathbf{x}) - P(c_2 \mid \mathbf{x}), \tag{7}$$

and $F(\mathbf{x}, \mathbf{w})$ be the system output.

Moreover, let us suppose that the system is trained in such a way that its response is $+1$ when $\mathbf{x}$ belongs to the class $c_1$, and $-1$ when $\mathbf{x}$ belongs to $c_2$:

$$F(\mathbf{x}, \mathbf{w}) = \begin{cases} +1 & \text{if } \mathbf{x} \in c_1 \\ -1 & \text{if } \mathbf{x} \in c_2 \end{cases} . \tag{8}$$

This is accomplished by mimizing (e.g., with the error backpropagation technique [26]) the *sample data error function*:

$$E_s(\mathbf{w}) = \sum_{\mathbf{x} \in X_1} [F(\mathbf{x}, \mathbf{w}) - 1]^2 + \sum_{\mathbf{x} \in X_2} [F(\mathbf{x}, \mathbf{w}) + 1]^2 . \tag{9}$$

We shall demonstrate that in the large $N$ limit, when $\mathbf{w}$ minimizes $E_s(\mathbf{w})$, then $\mathbf{w}$ minimizes also :

$$\epsilon^2(\mathbf{w}) = \int_\chi [F(\mathbf{x}, \mathbf{w}) - g(\mathbf{x})]^2 p(\mathbf{x}) d\mathbf{x}, \tag{10}$$

i.e., $F(\mathbf{x}, \mathbf{w})$ approximates $g(\mathbf{x})$.

Let us define the *average error function* as the function

$$E_a(\mathbf{w}) = \lim_{N \to \infty} \frac{1}{N} E_s(\mathbf{w}) \tag{11}$$

where $N$ is the total number of pattern vectors. $E_a(\mathbf{w})$ represents the error surface that is obtained when all possible vectors are used for the computation.

As assumed, in the large $N$ limit, we can suppose that the function $E_s$ is a reasonable approximation for $E_a$.

Let us rewrite the function $E_a$ as follows:

$$E_a(\mathbf{w}) = \lim_{N \to \infty} [\frac{n_1}{N} * \frac{1}{n_1} \sum_{\mathbf{x} \in X_1} [F(\mathbf{x}, \mathbf{w}) - 1]^2 + \frac{n_2}{N} * \frac{1}{n_2} \sum_{\mathbf{x} \in X_2} [F(\mathbf{x}, \mathbf{w}) + 1]^2] \tag{12}$$

where $n_i$ is the number of vectors belonging to the class $c_i$, and $\frac{n_1}{N}$ and $\frac{n_2}{N}$ represent, in the large $N$ limit, the a-priori probabilities $P(c_1)$ and $P(c_2)$, respectively.

The quantities $\frac{1}{n_i} \sum_{\mathbf{x} \in X_i} [F(\mathbf{x}, \mathbf{w}) - 1]^2$ represent the average value of $[F(\mathbf{x}, \mathbf{w}) - 1]^2$, over the patterns, provided that $\mathbf{x} \in c_i$.

By exploiting the strong law of large number [23], we can rewrite the Eq. 12 as

$$\begin{aligned} E_a(\mathbf{w}) &= P(c_1) \int_\chi [F(\mathbf{x}, \mathbf{w}) - 1]^2 p(\mathbf{x} \mid c_1) d\mathbf{x} + \\ &\quad + P(c_2) \int_\chi [F(\mathbf{x}, \mathbf{w}) + 1]^2 p(\mathbf{x} \mid c_2) d\mathbf{x} \\ &= \int_\chi [F^2(\mathbf{x}, \mathbf{w}) + 1][p(\mathbf{x} \mid c_1) P(c_1) + p(\mathbf{x} \mid c_2) P(c_2)] d\mathbf{x} - \\ &\quad 2 \int_\chi F(\mathbf{x}, \mathbf{w})[p(\mathbf{x} \mid c_1) P(c_1) - p(\mathbf{x} \mid c_2) P(c_2)] d\mathbf{x} \end{aligned} \tag{13}$$

The probability density function of the input vectors can be expressed as:

$$p(\mathbf{x}) = p(\mathbf{x} \mid c_1) P(c_1) + p(\mathbf{x} \mid c_2) P(c_2) \tag{14}$$

Moreover, by using the Bayes rule, we can write:

$$
\begin{aligned}
g(\mathbf{x})p(\mathbf{x}) &= [P(c_1 \mid \mathbf{x}) - P(c_2 \mid \mathbf{x})]p(\mathbf{x}) \\
&= P(c_1 \mid \mathbf{x})p(\mathbf{x}) - P(c_2 \mid \mathbf{x})p(\mathbf{x}) \\
&= p(\mathbf{x} \mid c_1)P(c_1) - p(\mathbf{x} \mid c_2)P(c_2)
\end{aligned}
\tag{15}
$$

Therefore:

$$
\begin{aligned}
E_a(\mathbf{w}) &= \int_\chi [F^2(\mathbf{x},\mathbf{w}) + 1]p(\mathbf{x})d\mathbf{x} - 2\int_\chi F(\mathbf{x},\mathbf{w})g(\mathbf{x})p(\mathbf{x}) \\
&= \int_\chi [F^2(\mathbf{x},\mathbf{w}) - 2F(\mathbf{x},\mathbf{w})g(\mathbf{x})]p(\mathbf{x})d\mathbf{x} + \int_\chi p(\mathbf{x})d\mathbf{x} \\
&= \int_\chi [F(\mathbf{x},\mathbf{w}) - g(\mathbf{x})]^2 p(\mathbf{x})d\mathbf{x} - \int_\chi g^2(\mathbf{x})p(\mathbf{x})d\mathbf{x} + \int_\chi p(\mathbf{x})d\mathbf{x} \\
&= \epsilon^2(\mathbf{w}) + \int_\chi [1 - g^2(\mathbf{x})]p(\mathbf{x})d\mathbf{x}
\end{aligned}
\tag{16}
$$

The learning algorithm will minimize $E_s$ with respect to $\mathbf{w}$. Then, as we assumed $E_s(\mathbf{w})$ to be a reasonable approximation for $E_a(\mathbf{w})$, the learning algorithm will minimize also $E_a$ with respect to $\mathbf{w}$. Moreover, $\int_\chi [1 - g^2(\mathbf{x})]p(\mathbf{x})d\mathbf{x}$ is a quantity that does not depend on $\mathbf{w}$, hence the optimization algorithm minimizes $\epsilon^2(\mathbf{w})$, too, which was to be demonstrated.

For the multiclass problem, let define the *mean square error* (MSE):

$$
MSE = \frac{\sum_{i,n}(y_i^n - t_i^n)^2}{N} \ , \tag{17}
$$

where $N$ is the size of the training set, $\mathbf{y}^n = (y_i^n)$ is the network output, and $\mathbf{t}^n = (t_i^n)$ is the n-th label of the associative pair of the training set. The components of $\mathbf{t}^n$ are defined as follows:

$$
t_i = \begin{cases} 1 & \text{if the pattern belongs to class } i, \\ 0 & \text{otherwise.} \end{cases} \tag{18}
$$

In the large training set limit, one can demonstrate that, if the MSE is assumed as the cost function, then when $\mathbf{w}$ minimizes the MSE, the system outputs $y_i$ approximate the Bayes optimal discriminant functions, i.e. the a posteriori class probabilities [25].

# 4 Learning in a FBFN Classifier

Exploiting the theory developed in the previous section for the multiclass problem, learning can be performed by minimizing the cost function shown in Eq. 17. This cost function can be minimized by many different techniques, among which the gradient descent technique, clustering methods [30], Kalman filters [12], genetic algorithms [6] , etc. In our experiments, the FBF network parameters (i.e., $m_{jk}$, $\sigma_{jk}$ and $\overline{y}_{ij}$) were obtained by performing a gradient descent with respect to the MSE across the training set. The learning formulas are [13, 30]:

$$
\Delta\overline{y}_{ij} = \eta_s[t_i - y_i]\phi_j \tag{19}
$$

$$
\Delta m_{jk} = \eta_m \phi_j \sum_i [t_i - y_i][\overline{y}_{ij} - y_i][x_k - m_{jk}]/\sigma_{jk}^2 \tag{20}
$$

$$
\Delta\sigma_{jk} = \eta_\sigma \phi_j \sum_i [t_i - y_i][\overline{y}_{ij} - y_i][x_k - m_{jk}]^2/\sigma_{jk}^3 \tag{21}
$$

where

$$
\phi_j = \frac{\prod_k \mu_{jk}(x_k)}{\sum_j \prod_k \mu_{jk}(x_k)} \tag{22}
$$

represents the normalized activation of rule $j$, and $\eta_s$, $\eta_m$, and $\eta_\sigma$ are the learning rates for $\overline{y}_{ij}$, $m_{jk}$, and $\sigma_{jk}$, respectively. In our experiments, we adopted an adaptive learning-rate scheme, as proposed in [27], and we noticed a considerable speed-up of the training phase [3] .

| Rules | Parameters | $(\%L, \%T)$ | epochs | (%L1, %T1) | Time (min) |
|---|---|---|---|---|---|
| 24 | 3312 | (97.55, 94.00) | 30 | (91.0, 88.0) | 3.5 |
| 32 | 4416 | (98.05, 95.00) | 30 | (89.0, 87.0) | 4.0 |
| 48 | 6624 | (98.65, 95.81) | 30 | (91.5, 88.5) | 16.0 |
| 64 | 8832 | (98.32, 96.02) | 30 | (93.5, 92.0) | 23.0 |
| 128 | 17664 | (98.97, 96.20) | 30 | (94.5, 93.5) | 46.0 |

Table 1: FBF network performances.From left to right, the columns represent the number of rules mapped on the FBF network, the number of adaptive parameters, the percentages of learning success (%L) and of test success (%T) after the training phase, the number of epochs required by the training phase, the percentages of learning success (%L1) and of test success (%T1) at end of the first epoch, and the duration of each epoch (in min on a Sun 10/20).

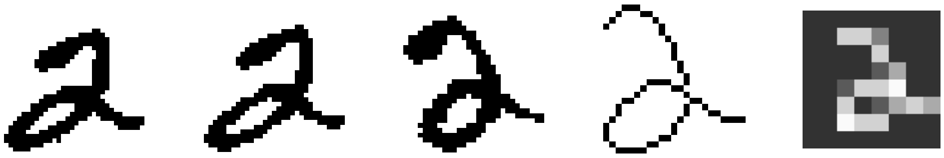
If a linguistic description of classes is available, in addition to the numerical training set, the FBF network permits the integration the these two types of information. On the contrary, if a linguistic description is not available, as is the case with our experiments, the structure identification must be achieved experimentally according to a performance-based criterion.

# 5 Data Base and Preprocessing

All the experiments reported in the following sections were carried out on a SUN 10/50 workstation (except those in Tab. 1). We used a data set of 30000 samples extracted from the NIST-3 data-base [9]. The NIST-3 data-base, distributed on a CD-ROM, contains 313389 handwritten characters coded as $128 \times 128$ binary-matrix images and labeled by the corresponding ASCII codes. We partitioned the data set in a training set, a test set and a validation set, each containing 10,000 associative pairs of segmented handwritten digits obtained from disjoint groups of writers. In order to improve the robustness of the classifier to noise and distortion, some preprocessing operations were performed. Among these there are some which try to increase the invariance of the classifier to scaling, shear, and thickness transformations. The following preprocessing steps were applied:

1. Digit image extraction from the CD-ROM and normalization to a $32 \times 32$ binary matrix.
2. Low-pass filtering in order to remove some small spots and holes from the image.
3. Application of a shear transform to the digit image to straighten the axis joining the first upper-left point of the digit image to the last lower-right point.
4. Image skeletonization [23], in order to be invariant to the thickness of the strokes.
5. Finally, transformation of the digit representation into a 64-element vector, each vector element representing the number of black pixels contained in adjacent $4 \times 4$ squares (local counting).

An example of application of the preprocessing is shown in Figure 2.



(a) (b) (c) (d) (e)

Figure 2: Preprocessing steps for a handwritten digit: Normalization (a), low-pass filtering (b), shear transform (c), skeletonization (d), local counting (e).

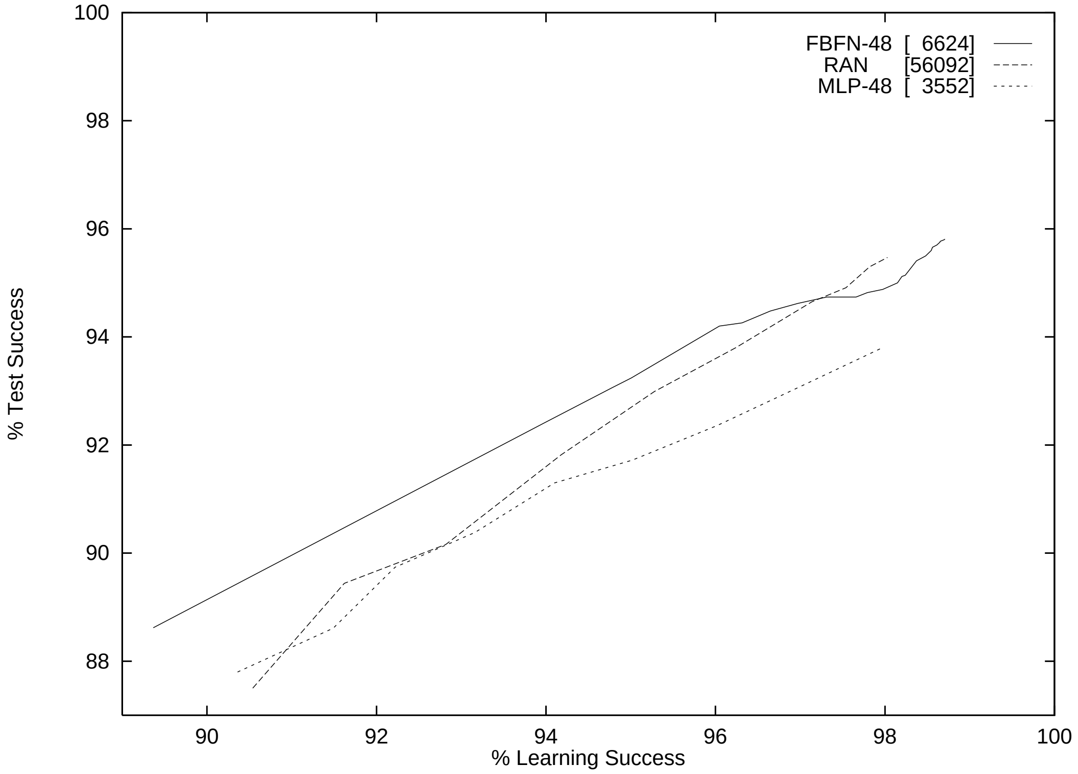



Figure 3: Comparison among the performances of the MLP, of the FBF network and of the RAN. The numbers within brackets refer to the parameters used by each of the three systems.

# 6 Classification Performances

In [18] a comparison is made between the performances of three connectionist feed-forward classifiers, namely a MLP [26], a Resource Allocating Network (RAN) [24], and a FBF network. The Resource Allocating Network is a Radial Basis Function neural network characterized by the growth of its architecture during the training phase [24]. The structure of each of the three systems was made up by 64 input units, 48 hidden nodes, and 10 output units (one for each class to be recognized). As shown in Figure 3, the three nets exhibit similar generalization properties, as we expected from a theoretical standpoint.

The MLP reaches a generalization value smaller than those of the other two learning machines. This may be due to the problem of *false positive*, as discussed by Lee [17]. The training phase of the FBF network is faster that that of the MLP. The RAN is the slowest one during the training phase; this depends on the growth of its architecture up to more than $50,000$ parameters (each of them to be optimized !). Moreover, it is worth to noting that the RAN shows the highest derivative of the Test Success, as compared with the Training Success. This depends on the possibility of allocating new units dynamically during the learning phase that characterizes this learning machine.

Tab. 1 gives the results of experiments on FBF networks with different number of rules. FBFNs show very good performances; for instance, in the case of an FBF network with 128 rules, a single epoch of 46 min is enough to obtain a percentage of test success equal to 93.5%.

# 7 FBFN Structure Identification and Semantic Phase Transition

In [4], ten different FBF networks have been trained on the training set described in Section 5. The FBF networks differ in the number of rules, ranging from 1 up to 10. In Tab. 2, for each class of digit the numbers of patterns in the training set and the percentage of patterns not correctly recognized by each FBF network at the end of the training phase, are shown. It is worth noting a close relationship between the number of rules in the FBF network and the number of classes which are recognized by the FBF network: $10-\beta$ classes are not recognized at all for FBF network with $\beta$ rules, while the other classes are well recognized. On the contrary, in the case of $FBF_{10}$ (or of FBF networks with more than 10 rules) the error rate is uniformly

| Training set | | $FBF_1$ | $FBF_2$ | $FBF_3$ | $FBF_4$ | $FBF_5$ |
|---|---|---|---|---|---|---|
| Class | Examples | % | % | % | % | % |
| 0 | 1052 | 100 | 100 | 100 | 100 | 100 |
| 1 | 1134 | 100 | 100 | 100 | 1.234 | 1.410 |
| 2 | 966 | 100 | 100 | 100 | 100 | 100 |
| 3 | 1059 | 100 | 100 | 100 | 100 | 100 |
| 4 | 967 | 100 | 3.177 | 1.964 | 2.378 | 3.309 |
| 5 | 842 | 100 | 100 | 100 | 100 | 100 |
| 6 | 948 | 100 | 100 | 100 | 100 | 4.00 |
| 7 | 1052 | 100 | 100 | 100 | 100 | 100 |
| 8 | 978 | 0.0 | 1.87 | 5.725 | 6.032 | 6.748 |
| 9 | 1002 | 100 | 100 | 2.295 | 2.694 | 2.694 |
| Training set | | $FBF_6$ | $FBF_7$ | $FBF_8$ | $FBF_9$ | $FBF_{10}$ |
| Class | Examples | % | % | % | % | % |
| 0 | 1052 | 1.615 | 1.711 | 1.711 | 1.711 | 1.806 |
| 1 | 1134 | 1.410 | 1.410 | 1.499 | 1.499 | 1.587 |
| 2 | 966 | 100. | 4.968 | 5.590 | 6.004 | 6.107 |
| 3 | 1059 | 100 | 100 | 100 | 3.966 | 4.438 |
| 4 | 967 | 3.826 | 3.826 | 3.826 | 3.826 | 3.826 |
| 5 | 842 | 100 | 100 | 100 | 100 | 11.63 |
| 6 | 948 | 4.00 | 4.113 | 4.113 | 4.113 | 4.535 |
| 7 | 1052 | 100 | 100 | 2.471 | 2.471 | 2.471 |
| 8 | 978 | 6.952 | 7.668 | 7.873 | 8.077 | 8.691 |
| 9 | 1002 | 2.894 | 3.093 | 3.393 | 3.493 | 3.592 |

Table 2: Training set error rates on 10,000 patterns by ten different FBF networks, ranging from 1 to 10 rules.

distributed along all the classes. This behavior of the FBF network, that we call *semantic phase transition*, has been confirmed by further series of simulations, and is not common in other feed-forward connectionist systems. The semantic phase transition phenomenon, gives a lower bound to the FBF network structure: the FBF network must contain a number of rules at least equal to the number of classes to be discriminated. The *semantic phase transition* observed for the FBF network can be explained by the *specialization* of each rule in a specific class. Even though it is hard to fully understand how a single rule of the system works, it is very easy to deduce the functional problem solved by each rule. Moreover, we have observed that the set of rules in a system with more than 10 rules can always be partitioned into 10 different subsets, each responsible for the classification of the examples of a specific class. Thus, even if the FBF network is organized as a supervised feedforward network, its behavior is closer to the one of a *competitive model* showing a strong specialization of the fuzzy rules. A pruning technique was proposed by Casalino et al. [5], in order to automatically remove less relevant fuzzy rules from an oversized system.

# 8 The Simplified FBF Network and Its Extension

For pattern recognition applications, from this FBF network a *Simplified FBF network* (SFBF network) can be obtained by assuming, in accordance with *rule specialization* [1]:

$$\overline{y}_{ij} \equiv \delta_{ij} = \begin{cases} 1 & \text{if rule } j \text{ is} \\ & \text{associated to class } i, \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

This assumption leads to both a system with as many units as classes and a strong simplification of the learning formulas, that become:

$$\Delta m_{jk} = \eta_m \phi_j \Upsilon_{ij}[x_k - m_{jk}]/\sigma_{jk}^2 \tag{24}$$

$$\Delta \sigma_{jk} = \eta_\sigma \phi_j \Upsilon_{ij}[x_k - m_{jk}]^2/\sigma_{jk}^3 \tag{25}$$

with

$$\Upsilon_{ij} = \begin{cases} (y_i - 1)^2 & \text{if } j = i \\ y_i^2 - y_i & \text{if } j \neq i \end{cases} \tag{26}$$

It is worth noting that, from Eq. 23 and the form of the defuzzifier, $y \in (0, 1)$ follows, and consequently

$$\Upsilon_{ij} = \begin{cases} \geq 0 & \text{if } j = i \\ \leq 0 & \text{if } j \neq i \end{cases} \tag{27}$$

holds.

Therefore, the learning rules of the SFBF network are competitive. During training, the means of the Gaussian membership functions of each rule move towards the patterns of the class associated to that rule, and escape from patterns belonging to other classes. At the same time, sigmas of Gaussian membership functions of each rule grow in order to increase the value of the membership function $\mu$ for patterns of the class associated to that rule, or shrink in order to reduce the value of the Gaussian membership function for patterns belonging to other classes. An interesting property of these new learning formulas is that they reduce by one order of magnitude the time of training.

One problem with the SFBF model is that, since it must have as many units as classification classes, it cannot be used for complex classification tasks. This constraint can be removed by introducing a new level of competition among units. The new defined network, called Extended Simplified FBF system (ESFBF), possesses $n_j$ units associated to each class $j$, for a total of $J = \sum_{j=1}^{I} n_j$ units. During learning the output of each unit is computed and the best unit for each class is selected, i.e., for each class $j$ the unit $i_j^* \in Idx_j = \{1, \ldots, n_j\}$ such that $i_j^* = \underset{i \,\in\, Idx_j}{arg\ max}\{\phi_i\}$ is selected. In that way the number of selected units is equal to the number of classes and the learning rules of the SFBF network can be applied. Thus, at each learning step, only the selected rules have the weights changed. During the operational phase, the input pattern is classified by the class label associated with the unit having maximum activity.

# 9 Performances of the SFBF and ESFBF networks

For comparison purposes, in Tab. 3 we have reported the performance of Nearest-Neighbor Rule (NR) [8], FBF networks with different number of units, SFBF network and EFBF with 20 rules on the validation set. In Figure 4 we have reported the performance of the $k$-Nearest-Neighbour for different values of $k$. Due to noise in the data, the best performance was obtained for $k = 1$. The training of the networks was stopped using the test set (early stopping).

We remark that the $FBF_{48}$ network got a better performance with respect to NR. Moreover, the average recognition time per pattern of the NR was .733 sec, while for the $FBF_{48}$ network it was .004 sec. In addition, it can be observed that the generalization performances of the $FBF_{10}$ network and the SFBF network were similar, while the SFBF network resulted to be faster in learning, in accordance with the lower complexity of the learning rules.

Finally, a significant improvement was obtained by the ESFBF with 20 rules (2 for each class). The training and test curves for this network are reported in Figure 5. It must be noted that the performance of this network was very close to the performance of the FBF network with 48 units, thus showing that the ESFBF networks, besides to be much faster in training (i.e., it is comparable with a FBF with 10 rules), can reach generalization performances which are comparable to the ones obtained by more complex models.

| MODEL | %S-Validation | Epochs | Epoch Duration (sec) |
|---|---|---|---|
| NR | 92.89 | – | – |
| $FBF_{48}$ | 94.09 | 13 | 1925 |
| $FBF_{12}$ | 92.26 | 36 | 307 |
| $FBF_{10}$ | 92.23 | 55 | 180 |
| SFBF | 91.36 | 10 | 30 |
| $ESFBF_{20}$ | 93.80 | 250 | 49 |

Table 3: Comparison among the NR, the FBF network (with 48, 12, and 10 rules), the SFBF network, and the ESFBF with 20 rule (two for each class). %S-Validation is the success rate on the validation set.

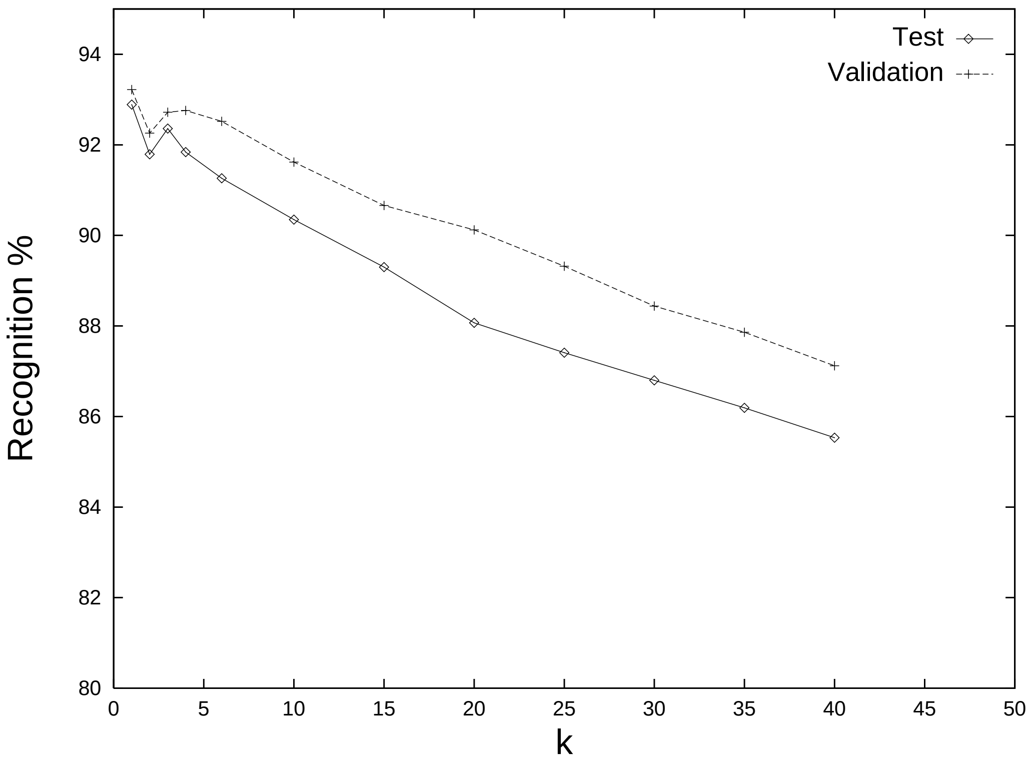



Figure 4: Performance of the k-Nearest-Neighbour on the test and validation sets using as reference database the training set.

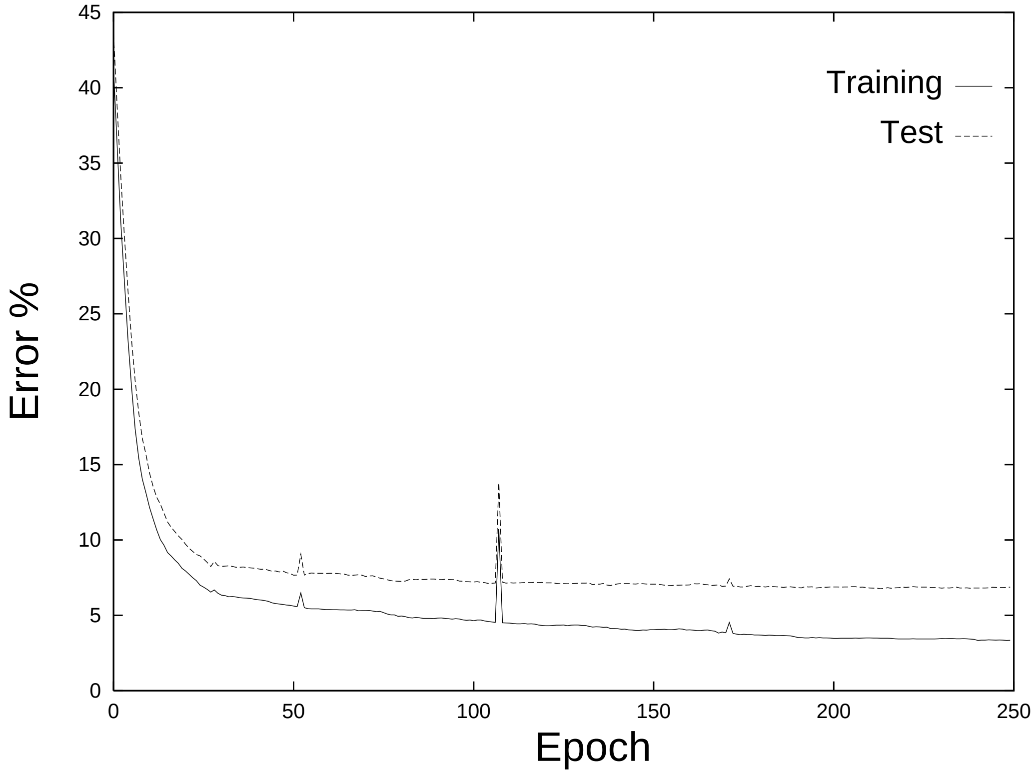



Figure 5: Training and test curves for a ESFBF with 20 units (2 units for each class).

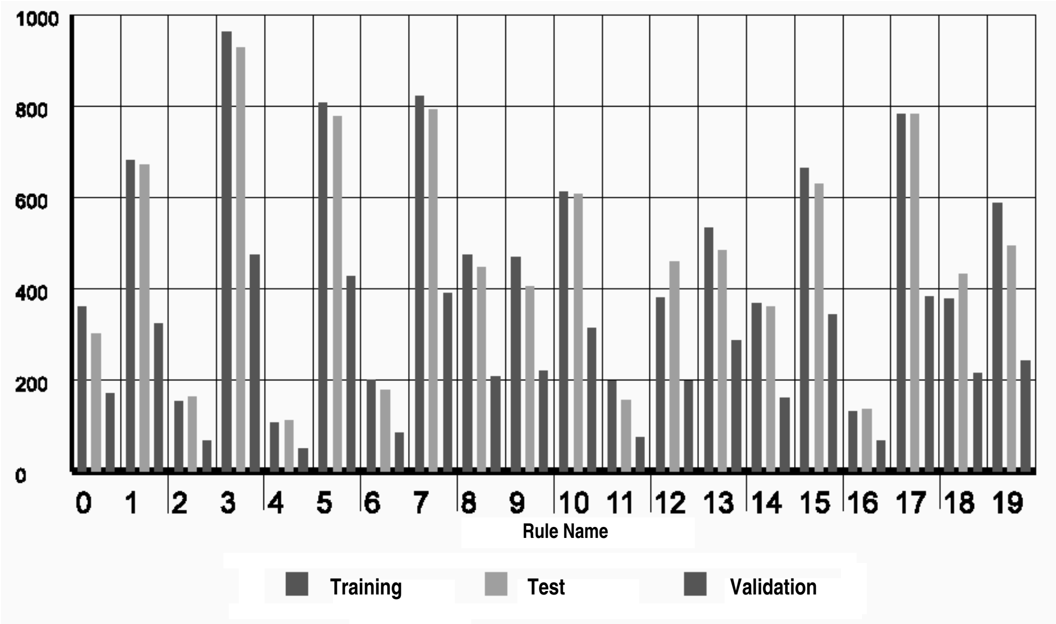



Figure 6: ESFBF system with 20 rules: histogram representing how many times a winning rule performs a correct classification. The rules are identified by numbers from 0 to 19: rules 0 and 1 are associated to class "0", rules 2 and 3 are associated to class "1", and so on. From the histogram it is clear that all the rules participate to the competition. The validation data refer to half of the validation set (i.e., 5000 patterns).

In Figure 6, we have reported an histogram representing how many times a winning rule performs a correct classification after training. This histogram is useful to understand whether, and to which extent, rules associated with the same class are used. From the histogram we deduce that all rules are used, even if in several cases there is one rule which is much more "active" than the other, which means that more patterns are covered by it.

# 10 Hybrid Network

An alternative way for recovering the loss in generalization exhibited by the SFBF network, consists in resorting to a hybrid pattern recognition scheme (HS) based on a hierarchy made up by an SFBF network with rejection, followed by a Nearest-Neighbor Rule classifier working on the patterns rejected by the SFBF network (see Figure 7). This approach is discussed here for the SFBF, however, it can be directly applied

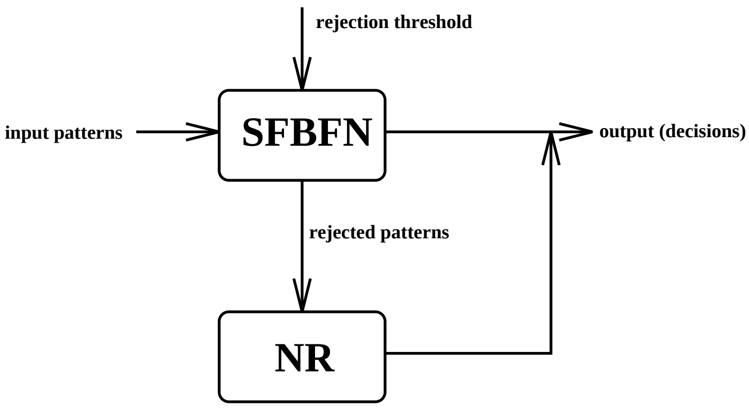



Figure 7: Hybrid Pattern Recognition Scheme combining the SFBF network and the NR.

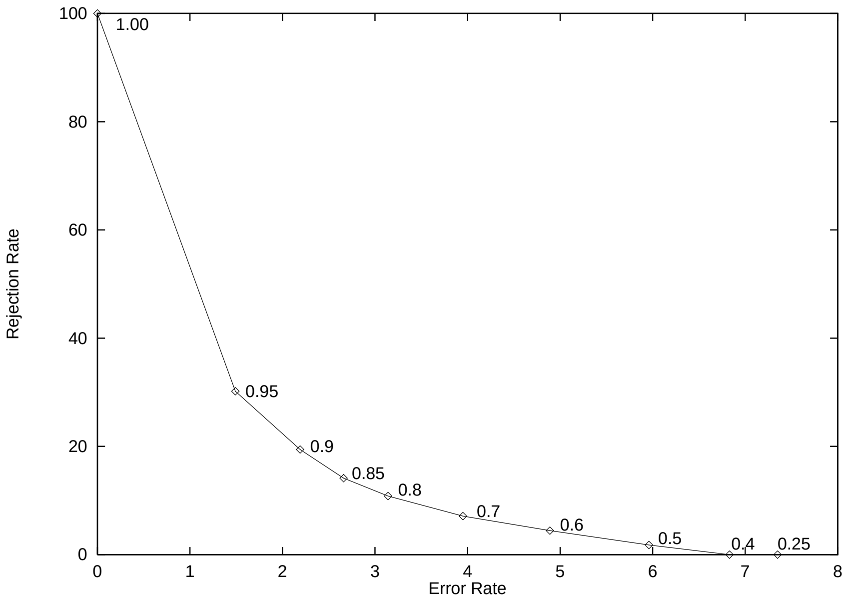



Figure 8: Effect of the rejections threshold (reported near each experimental point) on the training set.

to the ESFBF network as well, and experimental results, discussed elsewhere [10], confirm the improvement in performance also for this case.

After the training of the SFBF network, a rejection rule is implemented, consisting in a *rejection threshold* on the level of the higher output. If no output of the SFBF network is greater than the threshold, the pattern is rejected from the SFBF network and the Nearest-Neighbor Rule classifier is applied to it. By using the recognition threshold, the SFBF network classifies very quickly most of the patterns with small classification error, while a minority of patterns are forwarded to the NR for classification. For rejected patterns, the recognition speed depends mainly on the dimension of the space of search used by the NR.

In order to speed-up the recognition time of the NR classifier, we studied some optimizations of the hybrid scheme, consisting in editing strategies able to reduce the dimensions of the data-base to be used by the NR. The first optimization consisted in an *off-line editing strategy* to be used at the completion of the learning of the SFBF network. This method consists in a condensation of the training set, obtained by filtering the original training set by the SFBF network itself, with the application of the rejection threshold to the classification algorithm. By using this strategy, for each value of the rejection threshold on the SFBF network output, one obtains a *condensed data base* (CDB) containing only the patterns close to the decision surfaces, that are the most important for classification. In Figure 8, for the SFBF network, the rejection rate of patterns of the training set, versus the error rate on the accepted (classified) patterns, is plotted. Near each experimental point, the value of the corresponding rejection threshold applied to the SFBF network, is reported. This information is used for the selection of the most suitable threshold to be used within the hybrid scheme.

As shown in Tab. 4, the value of the rejection threshold (Rej-Thresh) affects the overall performance of the hybrid scheme. In this table CDB-Dim stands for the dimension of the resulting Condensed Data base, %E-Train is the error rate on the accepted patterns of the training set, %S-Valid is the success rate on the validation set, and Rec-Time is the average recognition time for the input pattern.

¿From Tab. 4, one could infer that, improvements of the recognition times with the hybrid scheme can be

| Rej-Thresh | CDB-Dimen | %E-Train | %S-Valid | Rec-Time (sec) |
|---|---|---|---|---|
| .98 | 4549 | .89 | 93.62 | .0750 |
| .95 | 3019 | 1.49 | 93.45 | .0324 |
| .70 | 713 | 3.95 | 92.43 | .0020 |

Table 4: Hybrid Pattern Recognition Scheme: Performances with different rejection thresholds (see text).

| L | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| %S-Valid | 93.46 | 93.53 | 93.68 | 93.58 | 93.64 | 93.66 | 93.61 | 93.60 | 93.45 |

Table 5: Hybrid Pattern Recognition Scheme: Performances with on-line edited Condensed Data Base (see text).

obtained to the detriment of the generalization rate, or vice-versa. However, at classification time one can use an efficient *on-line editing strategy*, by considering for the Nearest-Neighbor Rule (NR) only patterns belonging to classes that get the first L higher rates by the SFBF network.

In Tab. 5 for each value of L ($L \in [2, 10]$), the success rate on the validation set (%S-Valid) is shown. The rejection threshold for each experiment is set to .95. These experimental data point out that there is no significant dependence of the success rate on L. As a consequence a choice of $L = 2$, leads to good generalization performances and the higher recognition speed (about 5 time faster than for L=10, with our preliminary results).

In order to obtain a further speed-up in the search for the closest points by the NR, a parallel hardware can be developed, and/or the usage of optimized data structures, such as bumptrees [22], or randomized algorithms [7], can be experimented.

# 11 Conclusions

The universal function approximation results to be a key property for many connectionist systems, such as MLP, RBF and FLS. Moreover the choice of a suitable cost function for parameter identification permits to face efficiently pattern recognition problems.

In this framework, efficient supervised non-parametric fuzzy classifiers can be easily obtained, as shown in this chapter. The peculiarity of a classifier based on FLS, such as the FBF network studied here, consists in the fact that, if a linguistic description of classes is available in addition to a numerical training set, it can exploit both kinds of information for the pattern recognition task.

Moreover, we have shown that, according to the computational resources at hand and according to the target classification performances, it is possible to define a range of models, all derived from the original formulation of the FBF network, covering a large spectrum of controlling values involving training time, performance, and response time.

# Acknowledgments

This work was supported by grants from CNR, INFM, and MURST. We thank Nicola Giusti for implementing the competitive and hybrid algorithms.

# References

[1] D. Alfonso, F. Masulli, and A. Sperduti. Competitive learning in a classifier based on an adaptive fuzzy system. In P.G. Anderson and K. Warwick, editors, *Proceedings of the International ICSC Symposium on Industrial Intelligent Automation (IIA'96) and Soft Computing (SOCO'96), Reading, England*, pages C2–C8, Millet Alberta, Canada, 1996. ICSC.

[2] E. Barnard, F. Kanaya, and S. Miyake. Comments on 'Bayes statistical behavior and valid generalization of pattern classifying neural networks' (with reply). *IEEE Transactions on Neural Networks*, 3:1026–7, 1992.

[3] F. Casalino. Fuzzy systems for handwriting recognition. Laurea Thesis in Computer Science, University of Genoa, Genova - Italy, 1993. in Italian.

[4] F. Casalino, F. Masulli, and A. Sperduti. Rule specialization in networks of fuzzy basis functions. *Intelligent Automation and Soft Computing*, 4:73–82, 1998.

[5] F. Casalino, F. Masulli, A. Sperduti, and F. Vannucci. Semantic phase transition in a classifier based on an adaptive fuzzy system. In *Proceedings of the Third IEEE International Conference on Fuzzy Systems, IEEE-FUZZ94*, volume 2, pages 808–812, Orlando, FL, USA, 1994. IEEE.

[6] R. Caviglia. Soft-computing methods for time series forecasting. Laurea Thesis in Computer Science, University of Genoa, Genova - Italy, 1994. In Italian.

[7] K. L. Clarkson. A randomized algorithm for the closest-point queries. *SIAM Journal on Computing*, pages 830–847, 1988.

[8] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

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

[10] N. Giusti, F. Masulli, and A. Sperduti. Competitive and hybrid neuro-fuzzy models for supervised classification. In *Proceedings of 1997 IEEE International Conference on Neural Networks, INNC'97*, pages 516–519, Huston, USA, 1997. IEEE.

[11] J. Hampshire and B. Pearlmutter. Equivalence proofs for multi-layer perceptron classifiers and the Bayesian discriminant function. In D.S. Touretzky, G. Hinton, and T. Sejnowski, editors, *Connectionist Models: Proceedings of the 1990 Summer school*, pages 13–19, Denver, 1990. Morgan Kaufmann, San Mateo.

[12] J.S.R. Jang. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Trans. on Systems, Man, and Cybernetics*, 23:655–684, 1993.

[13] C.C. Jou. Comparing learning performance of neural networks and fuzzy systems. In *IEEE International Conference on Fuzzy Systems*, pages 1028–1033, San Francisco, 1993. IEEE, New York, NY.

[14] F. Kanaya and S. Miyake. Bayes statistical behavior and valid generalization of pattern classifying neural networks. *IEEE Transactions on Neural Networks*, 2:471–475, 1991.

[15] H.M. Kim and J.M. Mendel. Fuzzy basis functions: Comparisons with other basis functions. *IEEE Trans. on Fuzzy Systems*, 3:158–168, 1995.

[16] C.C. Lee. Fuzzy logic in control systems: fuzzy logic controller. I. *IEEE Transactions on Systems, Man and Cybernetics*, 20:404–418, 1990.

[17] Y. Lee. Handwritten digit recognition using k nearest-neighbor, radial-basis function, and backpropagation neural networks. *Neural Computation*, 3:440–449, 1991.

[18] F. Masulli. Bayesian classification by feedforward connectionist systems. In F. Masulli, P. G. Morasso, and A. Schenone, editors, *Neural Networks in Biomedicine - Proceedings of the Advanced School of the Italian Biomedical Physics Association - Como (Italy) 1993*, pages 145–162, Singapore, 1994. World Scientific. *(invited).*

[19] J.M. Mendel. Fuzzy logic systems for engineering: A tutorial. *Proceedings of the IEEE*, 83:345–377, 1995.

[20] S. Miyake and F. Kanaya. A neural network approach to a Bayesian statistical decision problem. *IEEE Transactions on Neural Networks*, 2:538–540, 1991.

[21] G.C. Mouzouris and J.M. Mendel. Non-singleton fuzzy logic systems. In *Proc. 1994 IEEE Conf. on Fuzzy Systems*, Orlando 1994, 1989. IEEE, New York.

[22] S. M. Omohundro. Bumptrees for efficient function, constraint, and classification learning. In R.P. Lippmann, J.E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 693–900. San Mateo, CA: Morgan Kaufmann, 1991.

[23] Theo Pavlidis. *Algorithms for Graphics and Image Processing.* Springer-Verlag, 1982.

[24] J. Platt. A resource-allocating network for function interpolation. *Neural Computation*, 3:213–225, 1991.

[25] D.W. Ruck, S.K. Rogers, M. Kabrisky, M.E. Oxley, and B.W. Suther. The multilayer perceptron as an approximation to a Bayes optimal discriminant function. *IEEE Transactions on Neural Networks*, 1:296–298, 1990.

[26] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 8, pages 318–362. MIT Press, Cambridge, 1986.

[27] T.P. Vogl, J.K. Mangis, A.K. Rigler, W.T. Zink, and D.L. Alkon. Accelerating the convergence of the back-propagation method. *Biological Cybernetics*, 59:257–263, 1988.

[28] L. Wang and J.M. Mendel. Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. *IEEE Trans. on Neural Networks*, 5:807–14, 1992.

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

[30] L. X. Wang. *Adaptive Fuzzy Systems and Control.* Prentice Hall, Englewood Cliffs, New Jersey, 1994.