PERGAMON

Pattern Recognition 35 (2002) 659–675

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Optimal design of radial basis function neural networks for fuzzy-rule extraction in high dimensional data

F. Behloul[a], B.P.F. Lelieveldt[a], A. Boudraa[b], J.H.C. Reiber[a,*]

[a]*Department of Radiology, Division of Image Processing, Leiden University Medical Center, P.O. Box 9600, Building 1 C2-S, 2300 Leiden RC, Netherlands*

[b]*L2TI Institut Galilee, Universite Paris 13, Avenue J.B. Clement, 93430 Villetaneuse, France*

Received 23 March 2000; received in revised form 28 December 2000; accepted 28 December 2000

Abstract

The design of an optimal radial basis function neural network (RBFNF) is not a straightforward procedure. In this paper we take advantage of the functional equivalence between RBFN and fuzzy inference systems to propose a novel efficient approach to RBFN design for fuzzy rule extraction. The method is based on advanced fuzzy clustering techniques. Solutions to practical problems are proposed. By combining these different solutions, a general methodology is derived. The efficiency of our method is demonstrated on challenging synthetic and real world data sets. © 2001 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Radial basis function networks; Fuzzy clustering; Fuzzy rule extraction; Neuro-fuzzy models; Adaptive network based fuzzy inference systems

1. Introduction

Rule learning is an increasingly important topic in both machine learning and data mining research. Machine learning concerns the development of algorithms or programs, which learn knowledge or skills while data mining is about the discovery of patterns or rules hidden in the data. Given a set of corresponding input–output values of a system, the challenge consists of identifying and formulating the relations between the input–output values in order to describe the system. To identify such relations, a functional input–output description may be provided. However, when dealing with complex processes, this is generally not feasible. One needs to look for alternative methods. The use of fuzzy models described through fuzzy rules has proven to be successful. Indeed, general knowledge about actions or conclusions can be expressed by a set of fuzzy if–then rules of a fuzzy inference system (FIS).

The basic structure of FIS consists of three conceptual parts: a selection of fuzzy rules (rule base), definitions of the membership functions used in the fuzzy rules (dictionary), and a reasoning mechanism, which performs the inference based on given facts to derive a conclusion (a fuzzy reasoning). In general, one designs a fuzzy inference system based on the past known behavior of the target system. The FIS is expected to reproduce the behavior of the target system, for example a human decision in a specific domain. Although the FIS model has a well-structured knowledge representation, it lacks the adaptability to deal with a changing external environment. The FIS is used only to mimic and not to learn and teach. If learning and automatic rule extraction abilities are required, then neural network concepts are preferred to fuzzy inference systems. Neural networks (NN) received the attention of the researchers because of their adaptivity and ability to learn. However, the semantic of a NN in terms of the problem to be solved is not explicit; the information is captured by a set of weights, and thus they are considered as black box systems.

Jang proposed a class of adaptive networks that is functionally equivalent to fuzzy inference systems [1]. He

* Corresponding author. Tel.: +31-71-526-3935/2138; fax: +31-71-526-6801.

E-mail address: reiber@lkeb.azl.nl (J.H.C. Reiber).

PII: S0031-3203(01)00033-4

demonstrated that under simple conditions, a radial basis function network (RBFN) is functionally equivalent to a FIS. While a FIS comprises a certain number of membership functions, a RBFN consists of radial basis functions. Both models produce a center-weighted response to small receptive fields, localizing the primary input excitation.

Under simple conditions, a FIS can be viewed as a neural network and vice versa. This hybridization[1] leads to a neuro-fuzzy model which cumulates the advantages of both models: the adaptivity of NNs and the well-structured knowledge of FISs. The functional equivalence provides a shortcut for better design of both FISs and RBFNs [2–7]. The analysis and learning algorithms for RBFNs are applicable to FIS and the fuzzy modeling procedure could be a good way of initializing a RBFN before training.

The use of fuzzy clustering for fuzzy rule extraction or design of a RBFN has been proposed in the literature [2,8,9]. One generally projects the fuzzy clusters in the domains of the variables leading to a grid partitioning of the domain space. However, this method may lead to a poor characterization of the system to be modeled because of the possible overlapping projections of different clusters. When an accurate model is desired the method tends to produce a large number of rules making the interpretability of the model difficult.

Fig. 1a shows a case problem where the cluster projections are highly overlapping, assuming that a clustering technique has been able to identify the different ellipsoidal clusters. In the case of axes-parallel ellipsoidal shapes the projection technique captures the information with a relatively good accuracy (Fig. 1b). However, the correlation between variables in some clusters may make the projection technique extract a large number of rules to model the system. Fig. 1c shows the rules one should get if the problem was limited to identifying the clusters c1, c2, and c5 (see Fig. 1a). In this case, the projection into the variable domains leads to six rules. Including clusters c2 and c3 makes the problem more difficult and will require a larger number of rules to identify c1, c2 and c5 (besides c2 and c3). The complexity of the problem further increase when dealing with high dimensional data.

In this work we propose to extract rules with multidimensional fuzzy sets since the full information about the data (clusters) may be in the entire space (see Fig. 1). Indeed, a multidimensional fuzzy set with arbitrary shape (hyper-ellipsoidal) can capture the characteristics of the clusters with a better accuracy. Thus, the system can be modeled with a smaller set of rules (one per cluster). Another motivation for the use of multidimensional fuzzy sets in fuzzy rules is related to the knowledge that one expects to get from an automatic training in high dimensions. For a human expert, a rule with a high number of input variables is very difficult to remember and thus, to learn. It is more intuitive to put a label on a multidimensional fuzzy set A and remember the rule "*if* x is A then y is B", than to put a label on each possible projection and learn several rules of type "*if* x_1 is A_1 and x_2 is A_2 and x_3 is A_3 and … x_n is A_n *then* y is B". In high dimensions, feature selection methods are used to reduce the dimensionality and thus the complexity of the problem [10,11]. In some applications this approach may be fruitful. However, the accuracy of the system using less features may drop for other applications. Furthermore, the reduced number on features may still be too high for rule extraction.

The idea of extracting rules using multidimensional fuzzy sets may sound as not new. Indeed, Delgado et al. proposed the characterization of fuzzy sets in the space dimension for a rapid prototyping for fuzzy rule-based modeling using fuzzy clustering [9]. However, because optimal characterization of multidimensional clusters is generally time consuming and presents some practical difficulties, researchers neglected it and preferred the use of simple clustering techniques as the well-known Fuzzy-C-Means algorithm. In Ref. [9] the authors justified the use of a simple clustering algorithm by the fact that only a good initial set of rules (rapid prototyping) was the aim of the work. However, they suggested to tune the extracted set of rules by genetic algorithms, which are well known to be computationally demanding.

In this paper we present a novel efficient approach to fuzzy rule extraction in multidimensional problems based on optimal clustering and neuro-fuzzy modeling. We propose solutions to the practical problems that are generally encountered using such modeling procedures, and give a general methodology on how to design an optimal neuro-fuzzy model for fuzzy rule extraction in high dimensional data. The method takes advantage of the recent advances in optimal fuzzy clustering. The use of appropriate similarity measures to group data into clusters is addressed, and cluster validity indices to define an optimal set of clusters to capture the information from the data are discussed. We put the rule extraction issue in a neuro-fuzzy framework in order to take advantage of the supervised training ability of NNs (RBFN) in order to tune the rules extracted by the unsupervised clustering technique.

The paper is organized as follows. In Section 2 we recall the functional equivalence between FIS and RBFN

[1] Hybrid (in contrast to combined) neuro-fuzzy models consist of homogeneous architectures that are usually neural network oriented (interpreting a FIS as NN), while a combined neuro-fuzzy model is a composed model of separate but cooperative NN and FIS.

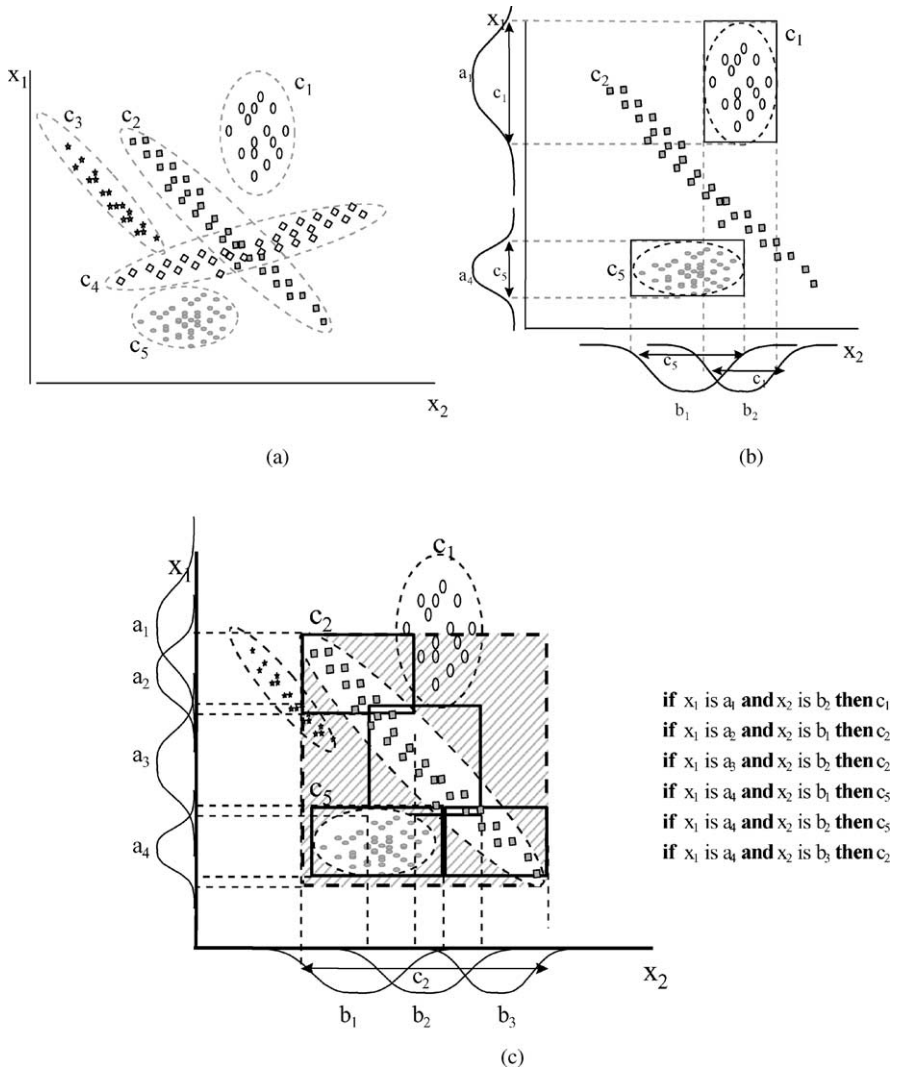

Fig. 1. Fuzzy rule extraction using projection and grid partitioning: (a) synthetic data presenting elongated and overlapping structures, (b) projection of axes-parallel ellipsoids, (c) projection of structures with correlated features.

and thus the interpretability of a RBFN in terms of fuzzy rules. In Section 3 we present the optimal clustering technique used to design the hidden layer of a RBFN and discuss the importance of the similarity measure to extract arbitrary hyper-ellipsoidal shaped radial basis functions (multidimensional membership functions). In Section 4 we give a concise description of the approach. The optimal number of rules is determined using cluster validity indices. Numerical examples and discussion are presented in Section 5 to illustrate the performance of the proposed method. Some conclusions and indications about the limitations of the proposed method are also provided. A summary of the approach is presented in Section 6.

2. Functional equivalence between FIS and RBFN

2.1. Fuzzy inference systems

A fuzzy if–then rule assumes the form: "If x is A then y is B" where A and B are linguistic values stipulated by fuzzy sets on the universes of discourse X and Y, respectively. A fuzzy set A is completely characterized by its membership function (MF) $\mu_A(x)$:

$$A = \{(x, \mu_A(x))/x \in X\}.$$

The MF $\mu_A(x)$ maps each element of X to a membership grade (or membership value) between 0 and 1. MFs

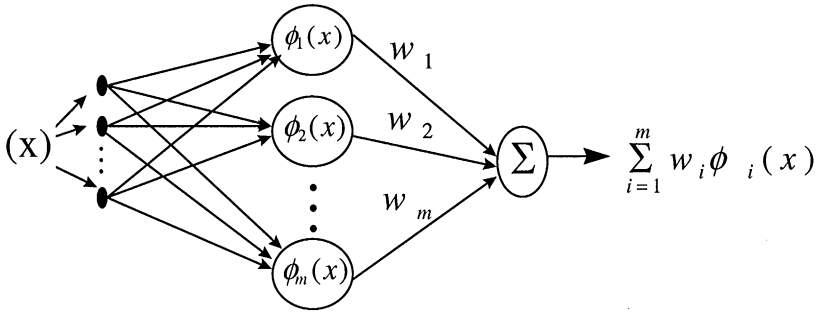

Fig. 2. Radial basis function network with one output node. Each component of the input vector x feeds forward to m radial basis function nodes whose output are linearly combined with weights $\{w_{i,\,i=1,\ldots,m}\}$ into the network output node.

can be defined by listing all the pairs $(x, \mu_A(x))$ or as a mathematical expression (preferably, when possible). "x is A" is called the antecedent or premise, while "y is B" is called the consequence or conclusion. The single rule with single antecedent is the simplest case. However, a fuzzy rule may have multiple antecedents. A fuzzy if–then rule with two antecedents is usually written as

"if x is A and y is B then z is C".

Multiple fuzzy rules with multiple antecedents can be involved in describing a system's behavior. Two types of FIS have been widely used in various applications: the Mamdani model [11] and Sugeno model [8]. The difference between the two models lies in the consequent part of their rules. The Mamdani FIS output is a fuzzy set and thus, a defuzzification step is required if a crisp output is desired. The Sugeno fuzzy model derives a crisp value. A typical fuzzy rule in the Sugeno model has the form: "if x is A and y is B then $z = f(x, y)$", where A and B are fuzzy sets in the antecedent, while $z = f(x, y)$ is a crisp function in the consequent. Usually, $f(x, y)$ is a polynomial in the input variables (x, y), but it can be any other type of function. When $f(x, y)$ is a first-order polynomial, the FIS is called a first-order Sugeno FIS. The zero-order Sugeno FIS corresponds to a zero-order polynomial $f(x, y)$. Since each rule has a crisp output, the overall output of the system is obtained by weighted average (or sometimes by a weighted sum).

2.2. Radial basis function neural networks

RBFNs have traditionally been associated with radial function networks in a single hidden layer [12]. The RBFN is a feed-forward neural network, which accomplishes an input–output nonlinear mapping by a linear combination of nonlinearly transformed inputs according to the following:

$$o_j = \sum_{i=1}^{m} w_i \phi_i(x) \tag{1}$$

or

$$o_j = \frac{\sum_{i=1}^{m} w_i \phi_i(x)}{\sum_{i=1}^{m} \phi_i(x)}, \tag{2}$$

where x is the input vector, o_j the output of the jth output node and w_i are the output linear combining weights. The $\phi_i(x)$ are radial basis functions (RBF) and m is the number of RBFs (see Fig. 2). An output of the network is a simple linear combination of the hidden neuron outputs (1) or the weighted average of the output associated with each receptive field (2).

Not all the RBFNs are the same. They may differ in the type of RBFs used and in training method and consequently they differ in performances. The most distinguishing feature of RBFs is that they are local, i.e. they give a significant response only in a neighborhood near a central point. Their response decreases monotonically with distance from a central point.[2] RBFs parameters are its center, its shape, and its width.

A typical local RBF is the Gaussian function. Centered at c and of width (or radius) r, it has the form

$$G(x, c, r) = \exp\left(\frac{\|x - c\|^2}{2r^2}\right). \tag{3}$$

The hidden and the output layers are generally trained sequentially: the radial basis function (RBF) parameters are first fixed and the optimal linear combining weights are then computed [13,14].

2.3. Functional equivalence

As pointed out by Jang in Ref. [1], FIS and RBFN seem to be rooted in the same soil. While the RBFN

[2] Strictly speaking, functions whose response increases monotonically away from a central point are also radial basis functions, but because they are not local, they are less interesting for classification applications.

consists of radial basis functions, the FIS comprises a certain number of membership functions. Both models produce a center-weighted response to small receptive fields. The conditions under which an RBFN and a FIS are functionally equivalent are given in the following:

1. The number of RBF units is equal to the number of fuzzy if–then rules in the FIS.
2. Each radial basis function of the RBFN is equal to a multidimensional Gaussian MF of the premise part of the corresponding fuzzy rule.
3. The output of each if–then rule is a constant value (a zero-order Sugeno model).
4. Both the RBFN and the FIS use the same aggregation method (weighted average or weighted sum) to derive their overall output.

Consider a two-input–one-output RBFN with a weighted average overall output, and two dimensional (2-D) Gaussian functions as RBFs.

$$\varphi_i(x, y) = e^{(-((x-ai)/2\sigma^2)^2 - ((y-bi)/2\sigma^2)^2)}, \quad (4)$$

where (a^i, b^i) is the center of the Gaussian and σ is its width.

A N-D Gaussian function presents the advantage of being composite, i.e. it can be decomposed into the product of N 1-D Gaussian functions:

$$\varphi_i(x, y) = e^{(-((x-ai)/2\sigma^2)^2 - ((y-bi)/2\sigma^2)^2)} = e^{-((x-ai)/2\sigma^2)^2} e^{-((y-bi)/2\sigma^2)^2}. \quad (5)$$

The product of two 1-D Gaussian functions can be interpreted as an AND operation: $\varphi_i(x, y)$ can be viewed as two statements joined by the connective AND (product):

"if x is A_i and y is B_i" $\Leftrightarrow \varphi_i(x, y) = \mu_{A_i}(x) \cdot \mu_{B_i}(y)$.

The corresponding FIS is a zero-order Sugeno FIS with 2 input variables, one output variable and 2 fuzzy rules, since each hidden node in the RBFN corresponds to a fuzzy rule.

The weights of the connections between the hidden layer and the output layer reflect a sort of impact factor of the corresponding rule in the whole inference system. In general, all the rules of a FIS are equally important. Their firing strength ($\varphi_i(x, y)$) will determine their relative contribution to the overall output of the system, for a given input. However, the weights w_i reflect the absolute contribution of the rule in FIS. In other words, the weights may be considered as the truth degree of a rule. Thus, owing to the functional equivalence, one may define automatically the truth value of a set of expert rules based on set of input–output example pairs. This can be realized using a RBFN training algorithm.

This functional equivalence makes the RBFNs (respecting the conditions given above) transparent networks that are able to explain what they learnt and how they make decisions (if used for a classification problem for instance). Therefore, automatic rule extraction issues can be tackled efficiently using RBFNs.

2.4. RBFN design

RBFNs have two distinct modifiable parts: the antecedent (hidden layer in RBFN) part and the consequent (output layer) part [13–17]. These two parts are usually adapted by two different optimization methods. The hidden layer definition is the most critical step in the design of an optimal RBFN. To determine its parameters, one has to decide on the number of neurons of the layer and their kernel functions (RBFs). A RBF is generally specified by its center and width. The simplest and most general method to decide the hidden layer number of neurons is to associate a neuron to each training pattern. However, for a large-scale data set this method is not convenient, especially when automatic rule extraction is involved. Therefore, a process of selecting a subset of basis functions from a large set of candidates is required.

In linear regression theory, subset selection is well known and one popular variant is *forward selection* in which the model starts empty ($m = 0$). Basis functions are selected one at a time and added to the network. The added RBF must reduce the sum of squared errors to the most. The process will stop adding RBFs when the error reaches a minimum and then starts to increase. In practice, heuristics are generally used. In such methods the way the patterns are selected affects the structure of the obtained network. Random selection is fast but sub-optimal while a sequential test of all the patterns is too time consuming. Orthogonal least squared learning (OLS) speeds up the forward selection [13]. An additional drawback concerns the network ability of generalization. By positioning RBF centers at some training pattern locations makes the network act as a nearest neighbor process. Indeed, a new pattern is compared to the centers and is assigned to the class of the nearest one. Furthermore, this approach tends to produce large networks with over-fitting problems.

An alternative way to determine the optimal RBFN configuration is to cluster the training patterns into groups according to some similarity measure and define a prototype for each group. A RBF node is assigned to each cluster. The typical methods to determine such clusters are the k-means or self-organizing feature maps. However, better clustering techniques are available in the literature and their use in the design of RBFNs still needs to be investigated.

3. Optimal RBFN design based on fuzzy clustering

The correspondence between basis function in RBFNs and membership functions in FIS provides an alternative way of designing RBFNs. Indeed, fuzzy clustering techniques are used to define automatically fuzzy membership functions of clusters, and thus can be applied to define automatically the parameters of the RBFs. An optimal clustering algorithm will therefore allow the design of an optimal RBFN architecture. By optimal clustering we mean automatic definition of the optimal number of clusters with the appropriate membership function shape and width. Since each neuron in the hidden layer corresponds to a rule, an optimal set of rules will be defined.

Our optimal RBFN design is based on: (i) the use of an appropriate similarity measure for fuzzy clustering: *the approximated exponential distance*, (ii) the use of a set of cluster validity indices to determine the number of hidden nodes (optimal number of clusters), and (iii) back-propagation for the training of the output layer after adequate initialization of the weights. The difficulties of each step and the corresponding solutions are presented in the following.

3.1. Fuzzy clustering and similarity measures

Bezdek introduced several clustering algorithms based on fuzzy set theory and an extension of the least squares error criterion [18]. Most analytical fuzzy clustering approaches are derived from Bezdek's fuzzy C-means (FCM) [19–21]. FCM has been proposed as an improvement over the earlier hard C-means clustering algorithm also known as the K-means algorithm.

FCM is a data clustering algorithm in which each data point belongs to a cluster to a degree specified by a membership degree. It partitions a collection of n data points $X_i\,(i=1,\ldots,n)$ into c fuzzy groups and finds a cluster center in each group, such that a cost function of a dissimilarity measure is minimized. The algorithm employs fuzzy partitioning such that a given data point can belong to several groups with a degree specified by membership grades between 0 and 1. A fuzzy c-partition of X is represented by a matrix $U=[\mu_{ik}]\in\Re^{c\times n}$, the entries of which satisfy the following constraints:

$$\text{(i)}\quad \mu_{ik}\in[0,1], 1\leqslant i\leqslant c, 1\leqslant k\leqslant n,$$

$$\text{(ii)}\quad \sum_{i=1}^{c}\mu_{ik}=1, 1\leqslant k\leqslant n,$$

$$\text{(iii)}\quad 0<\sum_{k=1}^{n}\mu_{ik}<n, 1\leqslant i\leqslant c. \tag{6}$$

U can be used to describe the cluster structure of X by interpreting μ_{ik} as the degree of membership of x_k to the cluster i. Good partitions U of X may be defined by the minimization of the following objective function [18]:

$$J_m(U,V)=\sum_{k=1}^{n}\sum_{i=1}^{c}(\mu_{ik})^m d_{ik}^2, \tag{7}$$

where $m\in[1,+\infty]$ is a weighting exponent called fuzzifier, $V=(v_1,v_2,\ldots,v_c)$ is the vector of the cluster centers, and d_{ik} is the distance between x_k and the ith cluster.

FCM Theorem (Bezdek [18]). *Assume* $m\geqslant 1$ *and* $d_{ik}^2>0$, $1\leqslant i\leqslant c$, $1\leqslant k\leqslant n$. (U,V) *may minimize* J_m *only if*

$$\mu_{ik}^{*}=\frac{1}{\sum_{j=1}^{c}(d_{ik}/d_{jk})^{2/(m-1)}}, \tag{8}$$

$$v_i^{*}=\frac{\sum_{k=1}^{n}(\mu_{ik})^m x_k}{\sum_{k=1}^{n}(\mu_{ik})^m}. \tag{9}$$

The FCM algorithm consists of iterations alternating between Eq. (8) and (9). This algorithm converges to either a local minimum or a saddle point of J_m [21].

FCM determines the cluster centers v_i and the membership matrix U for a given c value as follows:

Step 1: Initialize the membership matrix with random values between 0 and 1, such that constraints (i)–(iii) in Eq. (6) are satisfied.

Step 2: Calculate c fuzzy clusters centers v_i, $i=1,\ldots,c$ using Eq. (9).

Step 3: Compute the cost function according to Eq. (7). Stop if either it is below a certain tolerance value or its improvement over the previous iteration is below a certain threshold.

Step 4: Compute a new U using Eq. (8). Go to Step 2.

One of the major factors that influences the determination of appropriate clusters of points is the "(dis)similarity measure" chosen for the problem at hand. Indeed, the computation of the membership degrees μ_{ik}^{*} depends on the definition of the distance measure d_{ik}, which is usually a norm. Recent advances in fuzzy clustering have shown a spectacular ability to detect not only hypervolume clusters of different shapes [22–27], but also clusters that are actually thin shells such as curves and surfaces [29], by using an appropriate distance measure.

The inner product norms (quadratic norms) on R^n are generally used. The squared quadratic norm (distance) between a pattern vector x_i and the center of the kth cluster v_k is defined as follows:

$$d_{ik}^2=\|x_j-v_k\|_A=(x_j-v_k)^{\mathrm{T}}A(x_j-v_k), \tag{10}$$

where A is any positive definite $(N\times N)$ matrix.

Clusters found with inner product norms match smooth hyper-ellipsoidal shapes whose principal axes are determined by the eigenvectors of A. The identity matrix

is the simplest and most popular choice of A. The corresponding d_{ik} is the standard Euclidean distance and the corresponding fuzzy clustering algorithm is the so called FCM. The major drawback of the FCM algorithm is that it searches for hyper-spherical shaped clusters of approximately the same size; it has the undesirable property of splitting large elongated clusters. A different choice of A leads to clusters with a more or less hyper-ellipsoidal shapes. When A is a diagonal matrix with positive elements on the diagonal, the extracted clusters are axes-parallel hyper-ellipsoids. To extract clusters of arbitrary hyper-ellipsoidal shapes (not necessarily axis parallel), the covariance matrix is used [22,25,28]. Indeed, using the covariance matrix leads to a scaling distance in the principal component axes. Using the covariance matrix, one obtains the so called Mahalanobis distance.

For a fuzzy cluster, the classical covariance matrix is substituted by its fuzzy version defined by

$$F_k = \frac{\sum_{i=1}^{n} \mu_{ik}(x_i - v_k)(x_i - v_k)^{\mathrm{T}}}{\sum_{i=1}^{n} \mu_{ik}}. \tag{11}$$

According to our experience and to the literature [26], the use of the Mahanalobis distance makes the clustering algorithm tend to partition data into either very large or very small clusters. In order to avoid the extraction of very small clusters, Gustafson and Kessel [26] fix the size of each cluster a priori and proposed the following distance:

$$d_{GK}^2(x_i, v_k) = \alpha_k \det(F_k)^{1/2} \times D_M(x_i, v_k), \tag{12}$$

where α_k is the predefined constant which constrains the size of cluster k, and $D_M(x_i, v_k)$ is the Mahanalobis distance using the fuzzy covariance matrix:

$$D_M(x_j, v_k) = (x_j - v_k)^{\mathrm{T}} F_k (x_j - v_k).$$

The fact that the cluster volumes have to be specified a priori constitutes a major limitation of the proposed algorithm when no prior knowledge is available. Later in the literature, methods based on the maximum likelihood criterion have been proposed [26–28,30], which avoid the definition of constraints such as α_k. Gath and Geva proposed an elegant solution based on an "exponential" distance [22]:

$$d_{GG}^2(x_i, v_k) = \frac{\det(F_k)^{1/2}}{P_k} \times \exp\left(\frac{D_M(x_i, v_k)}{2}\right) \tag{13}$$

with

$$P_k = \frac{1}{n} \sum_{i=1}^{n} \mu_{ik}, \tag{14}$$

where P_k is the a priori probability of selecting the kth cluster.

The major advantage of the corresponding fuzzy clustering algorithm is obtaining good partition results in cases of great variability of cluster shapes (still hyper-elliptical), number of points and densities. Its major drawback however, is its strong sensitivity to the initial values of U and V matrices. Because of the exponential distance, the clustering algorithm seeks an optimum in a very narrow region and might be unstable during the iterative process of the clustering algorithm. Therefore, the FCM (with Euclidean distance) is used to initialize the proposed algorithm [22]. Clustering using this distance is called fuzzy maximum likelihood estimation (FMLE).

All the algorithms tracking hyper-ellipsoidal shaped clusters require the computation of the cluster covariance matrices and their corresponding inverses. The computation of a (fuzzy) covariance matrix requires a large number of training samples. However, in practice the covariance matrix is estimated from a finite number of training samples. It is particularly important to note that, when the ratio between the training sample size and the number of features is significantly small,[3] the covariance matrix becomes singular [30]. If the fuzzy covariance matrix is singular then the computation of the quadratic norm becomes theoretically impossible. In practice, one sets all the off-diagonal elements to zero so it becomes regular [14,15]. However, this is not always a good estimate of the covariance matrix and can lead to undesired partitions.

A second problem related to the use of the covariance matrices is related to the time consuming property of the corresponding clustering algorithms. Indeed, at each iteration, the covariance matrices and their corresponding inverses (and sometimes their determinant) have to be computed. This is computationally expensive in high dimensions with multiple clusters. This seems to be the reason why the Euclidean distance has been preferred in many applications.

In order to solve efficiently the problem of singularity of the covariance matrices and the time consuming property of the computation of the inverse, we suggest to use the Toeplitz covariance matrix estimator.

3.2. Toeplitz covariance matrix estimator

Let F be a covariance matrix. The components f_{ij} of the matrix may be expressed by

$$f_{ii} = \sigma_i^2, \quad f_{ij} = \rho_{ij}\sigma_i\sigma_j, \quad 1 \leqslant i \leqslant N \text{ and } 1 \leqslant j \leqslant N, \tag{15}$$

where σ_i^2 is the variance of x_i and ρ_{ij} is the correlation coefficient between x_i and x_j.

[3] According to the literature, a ratio smaller than three is considered significantly small.

Then $F = TRT$,

$$\text{where } T = \begin{bmatrix} \sigma_1 & & & 0 \\ & \sigma_2 & & \\ & & \ddots & \\ 0 & & & \sigma_N \end{bmatrix} \tag{16a}$$

and

$$R = \begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1N} \\ \rho_{21} & 1 & & \\ \vdots & & \ddots & \\ \rho_{N1} & & & 1 \end{bmatrix}. \tag{16b}$$

Thus $\det(F) = \det(T)\det(R)\det(T)$ and $F^{-1} = T^{-1}R^{-1}T^{-1}$,

$$\det(F) = \left(\prod_{i=1}^{N} \sigma_i\right)^2 \det(R),$$

$$T^{-1} = \begin{bmatrix} 1/\sigma_1 & & & 0 \\ & 1/\sigma_2 & & \\ & & \ddots & \\ 0 & & & 1/\sigma_N \end{bmatrix}. \tag{17}$$

The matrix R can be approximated by a particular form of a Toepliz matrix given below:

$$R_T = \begin{bmatrix} 1 & \rho & \cdots & \rho^{N-1} \\ \rho & 1 & & \\ \vdots & & \ddots & \rho \\ \rho^{N-1} & & \rho & 1 \end{bmatrix}, \tag{18}$$

$$R_T^{-1} = \frac{1}{1-\rho^2} \begin{bmatrix} 1 & -\rho & 0 & \cdots & 0 \\ -\rho & 1+\rho^2 & & \ddots & \vdots \\ 0 & & \ddots & & 0 \\ \vdots & \ddots & & 1+\rho^2 & -\rho \\ 0 & \cdots & 0 & -\rho & 1 \end{bmatrix} \tag{19}$$

and

$$\det(R_T) = (1-\rho^2)^{N-1} \tag{20}$$

The estimation process of a covariance matrix is as follows:

(i) Estimate the sample variance σ_i^2.
(ii) Estimate the sample covariance f_{ij} and divide $f_{i,i+1}$ by $\sigma_i\sigma_{i+1}$ to estimate $\rho_{i,i+1}$
(iii) Average $\rho_{i,i+1}$ over $i = 1, \ldots, N-1$, to obtain an estimate of ρ.
(iv) Use ρ to form R_T.

Note that the Toeplitz matrix requires only $(N+1)$ parameters, σ_i $(i = 1, \ldots, N)$ and ρ. Thus, its computational cost is not severe even if the dimension of the data is large.

The Toeplitz estimator has been used to estimate the covariance matrix in Ref. [31] for the design of a Parzen classifier. Two more kernel covariance estimators have been discussed: The Ness estimator and the orthogonal expansion estimator. The Toeplitz estimator has proven to be preferable to the others when the number of features is large or the number of training samples is small. Furthermore, it requires simple computations. The reader can refer to Refs. [32,33] for a presentation of Ness and orthogonal expansion estimator, respectively, and to Ref. [34] for a detailed presentation of the Toeplitz estimator. The comparison of the performances using an estimated covariance matrix against the exact one is not in the scope of this paper. Hamamoto et al. gave a comparison study in Ref. [31] in the case of Parzen classifier design, which is relatively close to the fuzzy clustering problem. They concluded that the performances were comparable.

In our approach, we use the Toeplitz estimator instead of the covariance matrix in Eq. (13) and refer to the distance as the *approximated exponential distance*.

3.3. Optimal number of clusters

Up to here the number of clusters was supposed to be known a priori. However, when this number cannot be defined according to some a priori knowledge, a cluster validity criterion is required in order to determine the optimal number of clusters [35–37]. Automatic rule extraction issues are generally dealing with a given set of input–output pairs. In pattern recognition problems the outputs are class labels. The true number of clusters is generally considered to be the number of labels present in the training data set. However, this number might not be optimal. Indeed, a class may be concave, i.e. group a number of separated overlapping clusters (see Fig. 3). When the visualization of the data is possible, that is, if the input space dimension is smaller than 4, one can identify these cases. However, in high dimensions this is not always possible. In this case, the use of cluster validity indices may solve the problem.

The criteria for the definition of an optimal partition of the data into subgroups are generally based on three requirements [37]:

1. Clear separation between the resulting clusters.
2. Minimal volume of the clusters

3. Maximal number of relevant data points concerned in the vicinity of the cluster centroid.

There is a number of cluster validation indices available in the literature. An analysis of several indices was performed by Pal and Bezdek in Refs. [38–39]. Tracking the optimal partition consists of varying the number of clusters between fixed minimum and maximum values and for each given number and computing the cluster validity index. An optimal partition corresponds to a minimum or a maximum (depending on the used index) value.

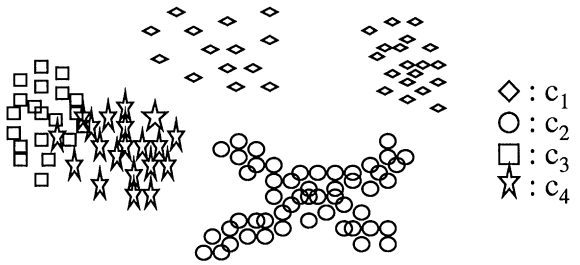

Fig. 3. Concave and overlapping classes. Class c1 is composed of two separate clusters and c2 is composed of two overlapping ellipsoidal clusters with different axis orientations. Classes c1 and c2 are concave clusters and an optimal clustering procedure would suggest to split them into different clusters. Classes c3 and c4 are overlapping classes that may be merged by an unsupervised clustering process.

There is no superior cluster validity index for all possible applications (data sets) with all possible combinations of the parameters of a clustering algorithm (the fuzzifier coefficient m in fuzzy clustering for instance). We suggest to use some of the most reliable ones (according to some experiences and to the literature) simultaneously and retain the optimal value delivered by the majority. In this work, 6 validity indices, presented in Table 1, have been used in the experiments.

The minimal value of the optimal number of clusters is set to the number of labels (classes) present in the training data. A smaller number may cause the merging of different classes that overlap into the same cluster. Thus, the optimal value may be smaller than the "true" number of clusters. Cluster validity indices organize data in an optimal number of groups based on "blind" requirements, that is, they are used in unsupervised techniques that do not take advantage of the labels of the patterns provided in the training data set (supervised training). The three criteria presented above may lead to merging neighboring classes. In an unsupervised frame it is impossible to avoid this merging when the three criteria are satisfied.

Table 1
Validity indices for optimal cluster-number identification

Validity index	Description	Optimal cluster no.
Hyper-volume	$V_{HV} = \sum_{i=1}^{c} [\det(F_i)]^{1/2}$	Minimum
Partition density	$V_D = \sum_{i=1}^{c} \frac{S_i}{[\det(F_i)]^{1/2}}$	Maximum
	$S_i = \sum_{j=1}^{N} u_{ij}$	
	$\forall x_j \in \{x_j/(x_j - v_i)F_i^{-1}(x_j - v_i) < 1\}$	
Average partition density (using the determinant of F_i)	$V_{AD} = \frac{1}{c}\sum_{i=1}^{c} \frac{S_i}{[\det(F_i)]^{1/2}}$	Maximum
Average partition density (using the trace of F_i)	$D_{AD} = \frac{1}{c}\sum_{i=1}^{c} \frac{S_i}{\mathrm{tr}(F_i)}$	Maximum
Trace of within cluster scatter matrix	$Vw = \mathrm{tr}(Sw)$	Maximum
	$Sw = \sum_{i=1}^{c} \left(F_i \sum_{j=1}^{N} u_{ij} \right)$	
Trace of between cluster scatter matrix	$V_B = \mathrm{tr}(S_B)$	Minimum
	$S_B = \sum_{i=1}^{c} \left(\sum_{j=1}^{N} u_{ij} \right)(v_i - v)\cdot(v_i - v)^{\mathrm{T}}$	
	$v = \frac{1}{c}\sum_{i=1}^{c} v_i$	

Therefore, we take advantage of the available labels: a penalty factor is added/subtracted to the validity index in case a cluster groups patterns belonging to different classes. The penalty factor reflects the variance of labels in the cluster. The higher the variance, the higher the penalty value. One can give different definitions for such a factor. In this paper we used the simple variance definition. The variance is added to the validity index when the minimum value represents the optimal number of clusters and it is subtracted to the index when a maximal value is desired.

3.4. Output layer supervised training and rule extraction

The remaining step in the optimal design of a RBFN is the determination of the weights of the output layer.

The training of the output layer is considered as a labeling phase. Indeed, the optimal clustering phase determines the best set of clusters to model the structure of the data and output layer is actually used to put the adequate label on clusters grouping patterns of a same class. In terms of fuzzy rules this layer determines the consequence parts of the extracted rules. Each output node represents a class.

In general, the output weights of a RBFN can be determined by a pseudo-inverse matrix which can be computationally demanding when the training set is large. In this work we used the delta-rule type of learning algorithm (Back-propagation) which is a less demanding technique.

The weight between a hidden node j and a output node i can be interpreted as the certainty degree to assign a pattern belonging to cluster j, to class i. A value of 1 means that cluster j groups patterns belonging exclusively to class i. A zero value means that no patterns from class I are present in cluster j. An intermediate value means that the cluster is heterogeneous reflecting a possible class overlapping (or merging). The weight values are thus considered as certainty values that can be defined as the percentage of patterns belonging to class i that were grouped in cluster j by the clustering algorithm. These values can be computed after the optimal clustering phase and be considered as initial set of weights for a back-propagation algorithm or as the final weights of the output layer.

4. Algorithm: elliptical radial basis function design

We have so far presented a general methodology for the optimal design of a RBFN for automatic fuzzy rule extraction; we refer to a RBFN designed by our approach as elliptical radial basis function network (ERBFN). In this section we recall the main steps in a concise algorithm for practical use of our approach.

Given a collection of n m-dimensional training data points $X_i\,(i = 1, \ldots, n)$ and their corresponding labels $l_i (i = 1, \ldots, n)$; $l_i \in \{1, \ldots, c\}$ where c is the number of classes. The issue is to extract rules in order to label correctly the data points. Let n_c be the number of clusters (RBFs), and c_{max} be the maximum value n_c can have.

Step 1: Set n_c to c and set the centers of the RBFs $\varphi_i(x)$ to the mean values of the c classes (one can chose to pick randomly a data point from each class to set the corresponding RBF).

Step 2: Unsupervised fuzzy clustering: First cluster the data using the FCM algorithm for few iterations (as described in Section 3.1), then apply the FMLE using the exponential distance. In relatively high dimensional problems one can use the Toeplitz estimator to compute the fuzzy covariance matrix and its inverse matrix and determinant, to reduce the computational time. The use of the Toeplitz estimator is required in case of singular covariance matrices.

Step 3: Compute the validity indexes as described in Section 3.3. If n_c is equal to the c_{max} go to step 4 else increase n_c by 1. The new RBF center can be picked randomly. However, we suggest to pick it from the most heterogeneous cluster and move the actual center of that cluster (by adding small random values or picking another data point randomly) and go back to step 2.

Step 4: Set the optimal number of nodes/rules $\boldsymbol{n_{opt}}$ to the number delivered by the majority of the validity indexes.

Step 5: Design a RBFN with $\boldsymbol{m}$ input nodes, $\boldsymbol{n_{opt}}$ hidden nodes and $\boldsymbol{c}$ output nodes. The centers of the hidden nodes are set to the prototypes of the clusters of the optimal partition. The shape of each radial function is defined by the (estimated) fuzzy covariance matrix (used in the distance computation). For each cluster (in the optimal partition), compute the percentage of each class, as described in Section 3.4. Set the weight of the connection between hidden node j and the output node i to the percentage of data points belonging to cluster j and labelled i. Test the obtained network. If further tuning is required, apply the well known backpropagation algorithm to train the network.

5. Experiments and discussion

In order to evaluate the efficiency of the proposed method, we performed experiments on synthetic and real world data. Cross validation technique is used to assess the performance of the classifiers. The results presented above are the average of 5 runs, where a random proportion of the data, 70% for synthetic data and 50% for the benchmark data (from each class), was used for training and the remaining data was used for testing.

5.1. Synthetic data experiments

The method was first tested on synthetic hyper-ellipsoidal clusters of varying size, axes orientation, position and density with normal distributions. By varying the distances between the cluster prototypes and controlling the variances of the features, cluster overlapping was controlled. Random fuzzy covariance matrices were generated in order to produce arbitrary oriented hyper-ellipsoids. The dimension, the number of clusters and the number of data points of each cluster were subjected to variation. Intuition about the system behavior was developed first on nonoverlapping hyper elliptical clusters and then by increasing the overlap between clusters belonging to the same class and finally by overlapping clusters belonging to different classes. The number of clusters per class varied from 1 to 3, and the total number of clusters varied from 2 to 10. The dimensions tested were $\{2, 3, \ldots, 10, 20, 40\}$.

In nonoverlapping cases the system performed very well; the accuracy (acc) was 100% with the number of hidden nodes equal to the number of clusters. The overlapping of clusters belonging to the same class did not affect the performance of the system. The number of hidden nodes was sometimes less than the number of clusters because overlapping clusters were merged. Since the merged clusters belong to the same class it did not affect the performances when the neighboring clusters from different classes were relatively far. However, as expected, the system was sensitive to the overlapping of the clusters belonging to different classes. We varied the cluster overlap from 10 to 30%; the performances varied from 94.1 to 72.3%, respectively. This behavior was predicted. In a totally overlapping area, the only way to distinguish the classes with acc = 100% is to assign a unit per pattern. However, this guaranties the correct classification of the trained patterns and not that of the test patterns.

In the following, we present two experiments to demonstrate the performances of our approach. For visualization convenience 2D-data sets are considered in the examples.

5.1.1. Experiment 1

Three elliptical classes with different variances and densities have been synthesized (see Fig. 4). Table 2 gives the description of the clusters. Classes C1 and C2 cross each other and present an important overlapping (24% of C1 overlaps with C2). Moreover, the centers of the classes are similar, what makes the Euclidean distance an inappropriate similarity measure. The performance of ERBFN is compared to that of the Gaussian RBFN (GRBFN). The parameters of the Gaussian functions are obtained after the FCM clustering step required for the initialization of the membership matrix (U) for EFCM. The cluster prototypes and variances are used to define the center and width of the Gaussian kernels, respectively.

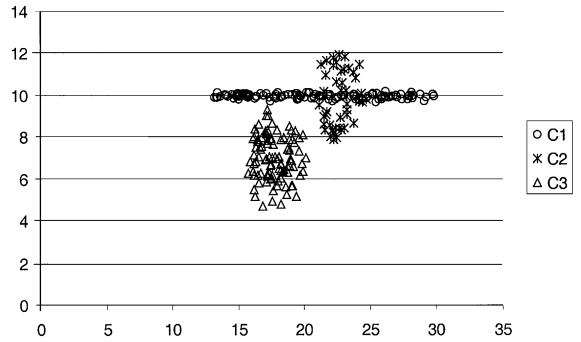

Fig. 4. Overlapping elliptical 2D data set used in experiment 1. Cluster C1 crosses cluster C2 and presents 24% overlapping.

Table 2
Data description of experiment 1

	Center (x, y)	Variance (x, y)	No. patterns
C1	(24, 10)	(24, 0.1)	123
C2	(24, 10)	(1, 2)	43
C3	(17, 7)	(1, 2)	102

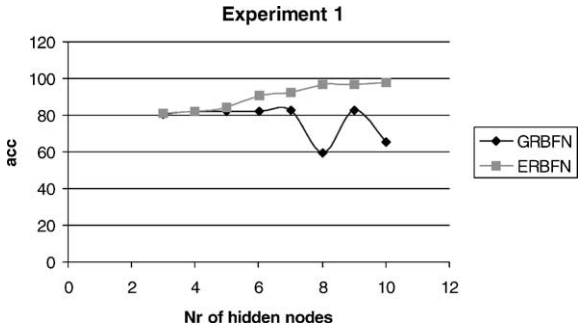

Fig. 5. GRBFN versus ERBFN in experiment 1. The data presents elongated (highly elliptical) clusters with important overlapping. In such cases ERBFN outperforms GRBFN.

Fig. 5 shows the accuracy of the GRBFN versus that of ERBFN. As one may note, the ERBFN showed better accuracy than GRBFN. Furthermore, ERBFN showed a growing accuracy according to the number of hidden units, while GRBFN's accuracy was oscillating. GRBFN were not able to accurately separate classes C1 and C3 in spite of the fact that these classes do not overlap. Furthermore, the crossing of C1 and C2 was not modeled by GRBFN. In Fig. 6, we show the optimal partition, according to the validity indices, for both FCM partitioning and FMLE partitioning. A number of seven clusters

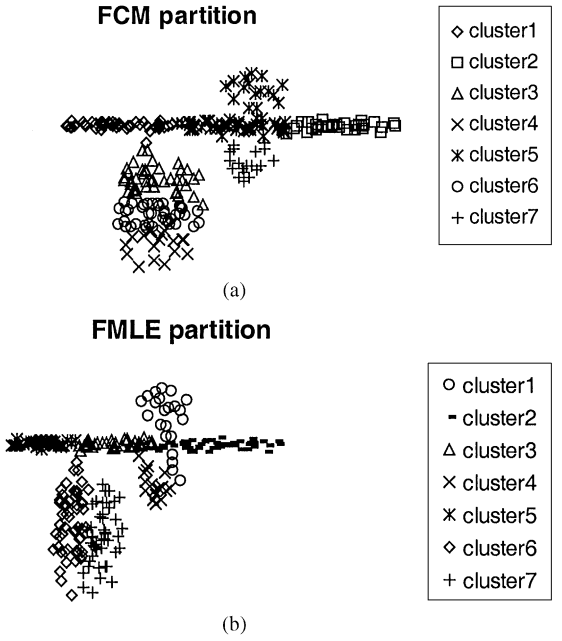

Fig. 6. Optimal clustering output in experiment 1: (a) partition delivered by the FCM algorithm, (b) partition delivered by the FMLE algorithm.

was a good compromise between the validity criteria and the homogeneity of the clusters. Fig. 6a shows that a part of C1 was merged with C2: cluster5 groups the top half of class C2 with the middle part of class C1. This reduces considerably the ability of the network to separate class C1 from class C2. Fig. 6b shows that ERBFN provided a better modeling of the overlapping classes C1 and C2. In this case 3 clusters (cluster2, cluster3 and cluster 5) are used to model class C1 and two clusters (cluster1 and cluster4) for C2. Class C3 has been captured by 2 cluster6 and cluster7.

One can notice that the optimal number of clusters 7 did not provide the best performance for ERBFN but for the GRBFN it did. One should keep in mind that cluster validity indices highly depend on the clustering algorithm. Different ways of partitioning lead to different optimal numbers of clusters. The optimal number of cluster is relative and not absolute. Since there are no best validity indices for all data sets and classifiers, it is difficult to define or choose appropriate ones. However, we noticed that the best performance of the networks was generally obtained using a number superior or equal to the "optimal" number of clusters (7 in this experiment) and is generally close to the "optimal" one (10 in this case). This is why cluster validity indices should be used to determine a first guess of optimal number of hidden nodes.

Experiments in higher dimensions have been carried out and ERBFN showed better performances over GRBFN in presence of elongated and close elliptical clusters.

5.1.2. Experiment 2

In this second example two banana shaped classes B1 (top) and B2 (bottom) were used to evaluate the method. Two data sets have been generated (see Fig. 7). In the first data set (banana1) B1 and B2 consisted of 100 patterns and in the second case (banana2) 1000 patterns per class. In both cases, a variance of 1 (around an arc) was used for each class. In the first set, B1 and B2 are adjacent but nonoverlapping while in the second set they present an important overlap. Fig. 8 gives the accuracy of the ERBFN and GRBFN. Very good accuracy was obtained using a small number of hidden nodes. Each class was split in several segments (clusters). In banana1 B1 was captured by 4 clusters (cluster2, cluster3, cluster5 and cluster7) and B2 was captured by 3 clusters (cluster1, cluster4 and cluster6). In banana2, B1 was captured by 4 clusters (cluster2, cluster4, cluster5 and cluster6) and B2 was captured by 3 clusters (cluster1, cluster3 and cluster7). Each cluster was modeled by a RBF node. The performances of GRBFN and ERBFN were comparable. The clusters automatically defined by fuzzy clustering are

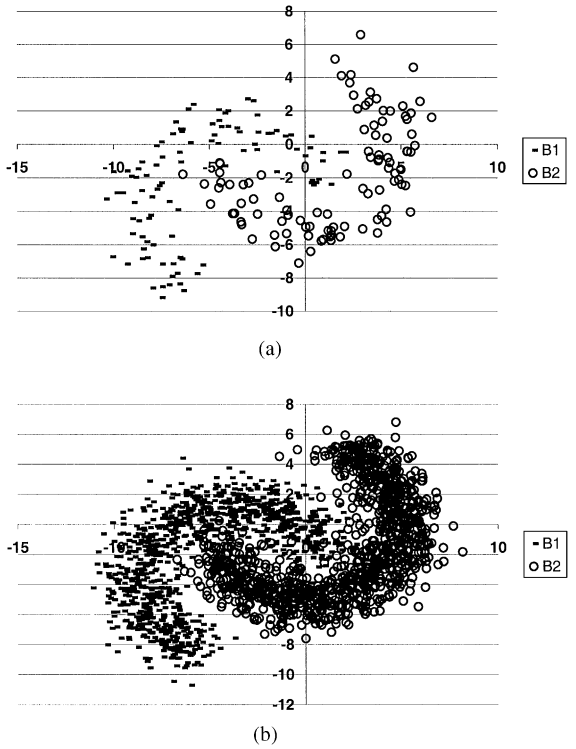

Fig. 7. Experiment 2: (a) banana1 data set: 100 patterns per class and no overlapping, (b) banana2 1000 patterns per class and overlapping.

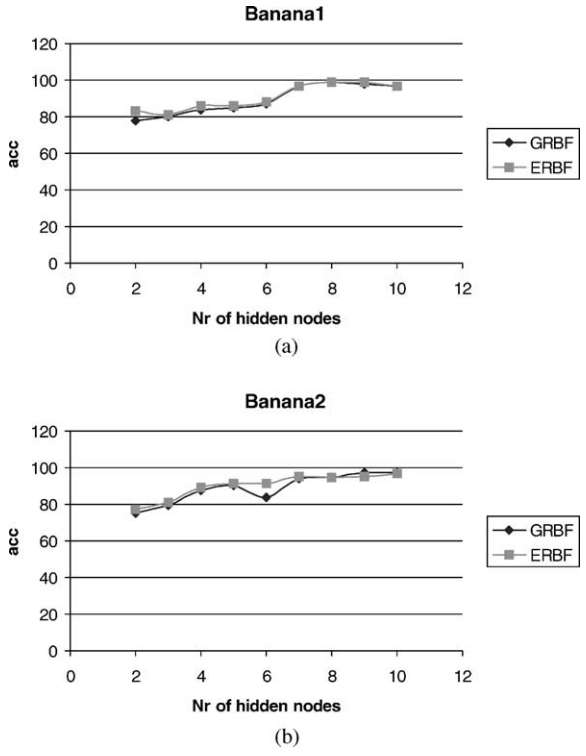

Fig. 8. GRBFN versus ERBFN in experiment 2 when applied on the banana1 data set (a) and when applied to the banana2 data set (b). The performances are comparable.

not highly elliptical and thus both kinds of RBFS can capture accurately the shape of the bananas using a set of circles or ellipses.

It is important to note that a small number of nodes (only 8 for banana1 and 10 for banana2) was sufficient to distinguish the two bananas with a very good accuracy (99% for banana1 and 96.8 for banana2). The optimal number of clusters was 7 for banana1 and 8 for banana2. As in experiment 1, the best performances were obtained using a number of nodes which is higher but still close. Thus our set of validity criteria was still relatively good since it gives an indication of the number of hidden nodes that should be used. The optimal number is around the value delivered by the majority of indices.

Extensive synthetic data experiments in higher dimensions have showed that in nonoverlapping cases the performances were comparable and in hyper-spherically shaped clusters the GRBFN showed a slightly better generalization than the ERBFN. However, for highly elliptically shaped clusters the ERBFN showed superior performance over GRBFN and the number of hidden nodes was generally smaller in the ERBFN than in the GRBFN. In fact, ERBFN can be viewed as being the result of the tuning of a GRBFN. Indeed, since FCM is used to define a first partition as an initialization step for UOFC, a GRBFN parameters can be derived without extra computational efforts. In high dimensions it is worth training both RBFNs when one does not know a priori the shape of clusters. If ERBFN performs better that GRBFN it implies that the data present highly elliptical clusters (elongated classes). However, if GRBFN does better it means that the data present hyper-spherical clusters or sparse elliptical clusters that can be modeled by a number of hyper-spheres (see Fig. 9).

5.2. Benchmark data experiments

Our focus lies on domains of continuous data in relatively high dimensions. The benchmark data selected for this study present non linear decision surfaces (or oblique) and noisy/overlapping data. Recently, Miroslav Kubat [40] presented interesting results on benchmark data obtained using RBFN. He used decision trees to initialize the RBFs centers and widths and derived analytically the weights of the output layer by means of a pseudo-inverse matrix to train the networks. His network is referred to as tree based radial basis function network (TB-RBFN). He compared the performances of a RBFN, initialized using decision trees, to the performances of the decision trees generated by Quinlan's C4.5. In this paper we compare the performances of our ERBFN to the performances given by Kubat [40].

The public-domain benchmark data were selected from the University of California, Irvine, repository [41]: Liver disorder (bupa), diabetes (diab), glass and vehicles. Information about the data is given in Table 3. The rightmost column gives the percentage with which the majority concept is represented [40]. This number represents the performance that would be achieved if the system consistently labeled all testing examples with the label of the most frequently represented concept.

Table 4 presents the results obtained by Kubat using TB-RBF and C4.5 and the results we obtained for the same number of units. Table 5 presents the best results obtained by our approach. The number of rules per class is given in Table 6.

For the Bupa data set we obtained comparable results when using the same number of units (24). TB-TBF show a slightly better performance (2.53%). However, ERBFN showed a better accuracy when increasing the number of units. A significant improvement was obtained with 54 rules. GRBFN had the same performances and increasing the number of clusters (node) did not show any significant improvement. Thus, this data must present elongated structures.

In Diabet, ERBFN showed the best performances while GRBFN showed comparable performances with those of TB-RBFN when using the same number of hidden units. ERBFN showed significant improvements (4.62%) using 55 units instead of 42. This data presents

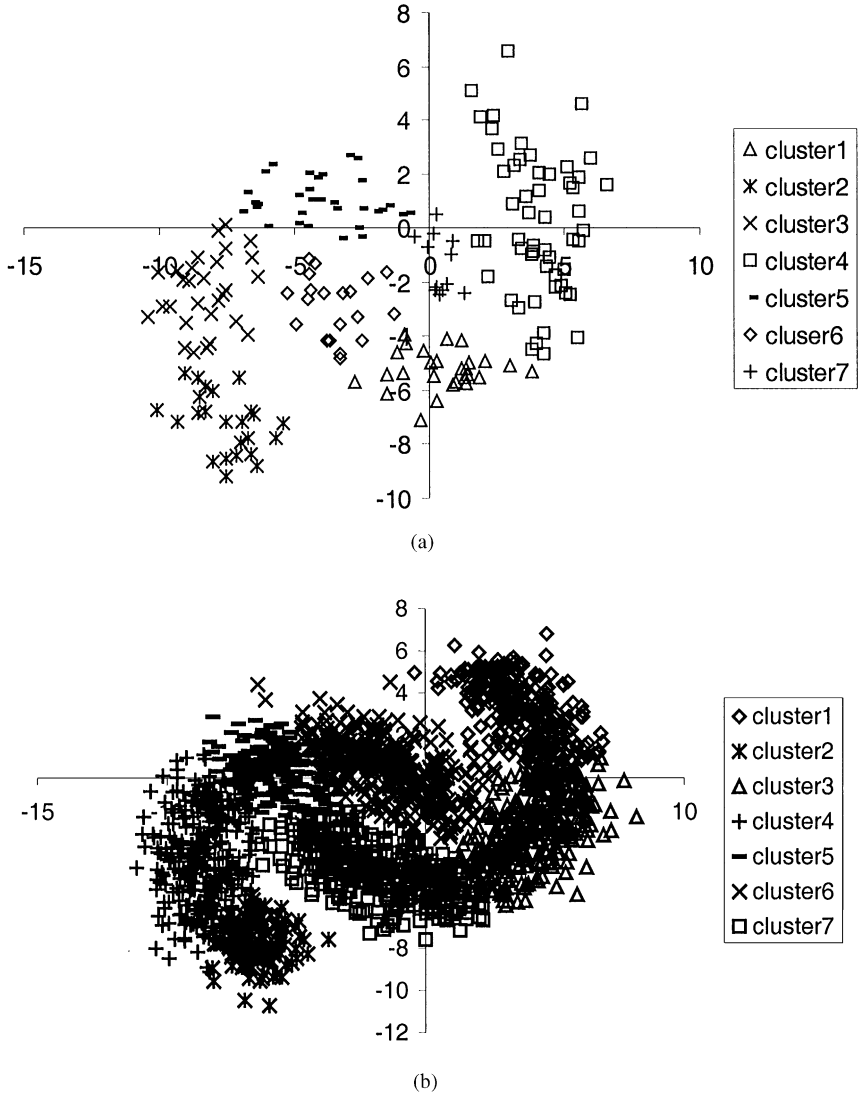

Fig. 9. Optimal clustering output (a) for banana1 data set, (b) for banana2 data set.

Table 3
Benchmark data sets

	Dimension	No. classes	No. examples	Majority
Bupa	6	2	345	58.0
Diabet	8	2	768	65.1
Glass	9	6	214	35.5
Vehicles	18	4	846	28.3

concave classes and hyper-elliptical (elongated) structures, since ERBFN did better than GRBFN and the accuracy increased when the number of nodes was increased. With 55 nodes (rules) the ERBFN presented the best accuracy reported in the literature for the Diabet dataset.

The glass data set presented the most spectacular results. ERBF showed a significant improvement (5.77%) over TB-RBFN while using the same number of hidden units. GRBFN presented comparable accuracy. Significant improvement is obtained with 21 hidden units by both RBFNs (12.5% for GRBFN and 10.53 for ERBFN). The data does not seem to present elongated structures since GRBFN and ERBFN performances are nearly identical. However, both RBFNs did not succeed to distinguish class c3 (see Table 6). No node (rule) is dedicated to this class. This is because few patterns (17) represent this class and it seems that it has an important

Table 4
Performances using the same number of hidden units as those derived for the TB-RBF and C4.5 approaches

Data set	GRBF		ERBF		TB-RBF		C4.5
	No. units	acc	No. unit	acc	No. unit	acc	acc
Bupa	24	66.27	24	62.2	24.0	68.8	65.3
Diabet	42	74.74	42	77.34	42.6	74.8	71.8
Glass	16	64.76	16	68.57	16.4	62.8	61.7
Vehicles	57	74.29	57	63.27	57.2	74.6	64.9

Table 5
Optimal number of hidden nodes

Data set	GRBF		ERBF	
	No. units	acc	No. units	acc
Bupa	53.0	66.86	54	73.25
Diabet	39.0	75.78	55	79.42
Glass	21.0	75.3	21	73.33
Vehicles	57.0	74.29	57	63.27

Table 6
Number of rules (clusters) per class for each data set

	No. nodes	C1	C2	C3	C4	C5	C6
Bupa	54	36	18	—	—	—	—
Diabet	55	36	19	—	—	—	—
Glass	21	5	6	0	4	2	4
Vehicles	57	19	15	8	15	—	—

overlap with other classes C1 and C2. Class C4 groups less patterns (13) and so does class C5 (9). These two classes are still distinguishable, thus, they do not present overlap with other classes.

In the vehicle data set, GRBFN presented comparable results to TB-RBFN however ERBFN did not show any improvement compared to C4.5. This brings back in mind the known drawback of FMLE algorithm: It can seek an optimum in a narrow local region and may show some time instability.

Our approach was successful in most of the experiments we carried out. Besides the design of a good classifier one can derive some information about structure of the data in high dimensions (presence of convex or concave classes, presence of overlapping, the shape of the classes or their segments).

6. Summary

In this paper an efficient method to design RBFNs in high dimensional data sets for rule extraction is presented. Rules are extracted in the entire feature space in order to model the eventual correlation between the feature (arbitrary shaped and oriented classes). The method takes advantage of the functional equivalence between RBFNs and FISs pointed out by Jang. Advanced fuzzy clustering is used to design an optimal RBFN. Optimality concerns the smaller number of hidden nodes possible with adequate shapes of kernel functions with high accuracy. Moreover, there is a straightforward mapping of fuzzy clustering output to RBF nodes: one node per cluster, RBF centers correspond to the cluster prototypes and the shape and width of the kernels are determined by the fuzzy covariance matrix of the corresponding cluster.

Optimal fuzzy clustering allows optimal RBFN design, although it is well known that clustering may be sensitive to the type of similarity measure used to group the patterns. In this paper we suggested the use of the estimated exponential distance, which is an approximation of the distance proposed by Gath and Geva [22]. Our distance is based on the Toeplitz covariance matrix estimator. This makes the distance insensitive to matrix singularity and does not involve inversion of the covariance matrix. Indeed the inverse and determinant of the Toeplitz estimator are analytically known. Hyper-elliptical radial basis functions are then used to model the extracted elliptical clusters.

Since there is no best validity index, the optimal number of neurons (= clusters or rules) is defined using several validity indices. The optimal number is the one suggested by the majority in a voting procedure. Furthermore, in order to avoid blind clustering, a penalty factor is added to the validity index, reflecting the heterogeneity of labels in clusters. The computation of such a factor is possible because RBFN are supervised neural networks, thus, the exact labeling of training patterns is provided a priori. The higher the variance of labels, the higher the penalty values. The "true" number of cluster may be higher than the number of classes in case of concave classes. The optimal fuzzy clustering can identify concave structures. A concave class may be split in several convex clusters and thus be modeled by several convex hyperspherical or hyper-elliptical kernels. Since each cluster corresponds to a RBF node and thus to a fuzzy rule,

optimal set of rules describing the structure of the data in the entire space is extracted.

In this paper, efficient and practical solutions to known problems encountered while designing RBFN networks are presented and a general methodology is presented. The performances of the method are demonstrated on challenging synthetic and real world data sets. The focus lie on domains of continuous data presenting nonlinear decision surfaces and noisy and/or overlapping areas. For each example presented in this paper small networks were derived to model the data with good accuracy. The structure of the data can be identified and described by few fuzzy rules defined in the entire space, which is more practical for high dimensional data sets.

References

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

[2] R.T. Yager, D.P. Filev, Unified structure and parameter identification of fuzzy models, IEEE Trans. System Man Cybernet. 23 (1993) 1198–1205.

[3] J. Nie, D.A. Linkens, Learning control using fuzzified self-organizing radial basis function network, IEEE Trans. Fuzzy Systems 1 (4) (1993) 280–287.

[4] C.F. Juang, C.T. Lin, An on-line self-constructing neural fuzzy inference network and its applications, IEEE Trans. Fuzzy Systems 6 (1) (1998) 12–32.

[5] F. Klaw, R. Kruse, Constructing a fuzzy controller from data, Fuzzy Sets and Systems 85 (1997) 177–193.

[6] K.M. Lee, D.H. Kwak, H. Leekwang, Tuning of fuzzy models by fuzzy neural networks, Fuzzy Sets and Systems 76 (1) (1995) 47–63.

[7] Z.Q. Liu, F. Yan, Fuzzy neural network in case-based diagnostic system, IEEE Trans. Fuzzy Systems 5 (2) (1997) 209–222.

[8] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. System Man Cybernet. 15 (1985) 116–132.

[9] M. Delgado, A.F. Gomez-Skarmeta, F. Martin, A fuzzy clustering-based rapid prototyping for fuzzy rule-based modeling, IEEE Trans. Fuzzy Systems 5 (2) (1997) 223–233.

[10] S.J. Raudys, A.K. Jain, Small sample size effects in statistical pattern recognition: recommendations for practitioners, IEEE Trans. Pattern Anal. Mach. Intell. 13 (3) (1991) 252–264.

[11] M. Ramze Rezaee, B. Goedhart, B.P.F. Lelieveldt, J.H.C. Reiber, Fuzzy feature selection, Pattern Recognition 32 (1999) 2011–2019.

[12] D.S. Broomhead, D. Lowe, Multivariable function interpolation and adaptive networks, Complex Systems 2 (1988) 321–355.

[13] S. Chen, C.F.N. Cowan, P.M. Grant, Orthogonal least square learning algorithm for radial basis function networks, IEEE Trans. Neural Networks 2 (2) (1991) 302–309.

[14] Y.S. Hwang, S.Y. Bang, An efficient method to construct a radial basis function neural network classifier, Neural Networks 10 (8) (1997) 1495–1503.

[15] M.J.D. Powell, Radial basis functions approximations to polynomials, Proceeding of 12th Biennial Numerical Analysis Conference, Dundee, 1987, pp. 223–241.

[16] B. Mulgrew, Applying radial basis functions, IEEE Signal Process. Mag. (1996) 50–64.

[17] M. Musavi, W. Ahmed, K. Chan, K. Faris, D. Hummels, On the training of radial basis function classifiers, Neural Networks 5 (4) (1992) 595–603.

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

[19] S. Abe, R. Thawonmas, A fuzzy classifier with elliptical regions, IEEE Trans. Fuzzy Systems 5 (1997) 358–368.

[20] S. Abe, R. Thawnmas, M. Kayama, A fuzzy classifier with ellipsoidal regions for diagnosis problems, IEEE Trans. System Man Cybernet. Part C: Appl. Rev. 29 (1) (1999) 140–149.

[21] M.S. Yang, Convergence properties of the generalised fuzzy-C-means clustering algorithms, Comput. Math. Appl. 25 (12) (1993) 3–11.

[22] I. Gath, A.B. Geva, Unsupervised optimal fuzzy clustering, IEEE Trans. Pattern Anal. Mach. Intell. 11 (7) (1989) 773–781.

[23] R.N. Davé, R. Krishnapuram, Robust clustering methods: a unified view, IEEE Trans. Fuzzy Systems 5 (2) (1997) 270–293.

[24] L. Bobrowski, J.C. Bezedek, C-means clustering with the L1 and L^{∞} norms, IEEE Trans. System Man Cybernet. 21 (3) (1991) 545–554.

[25] P.J. Rousseeuw, L. Kaufma, E. Trauwaert, Fuzzy clustering using scatter matrices, Comput. Statist. Data Anal. 23 (1996) 135–151.

[26] D.E. Gustafson, W.C. Kessel, Fuzzy clustering with a fuzzy covariance matrix, IEEE CDC, San Diego, 1979, pp. 761–766.

[27] P.J. Rousseeuw, E. Trauwaert, L. Kaufma, Fuzzy clustering with high contrast, J. Comput. Appl. Math. 64 (1995) 81–90.

[28] I. Gath, A.B. Geva, Fuzzy clustering for the estimation of the parameters of the components of mixtures of normal distributions, Pattern Recognition Lett. 9 (1989) 77–86.

[29] R.N. Davé, R. Krishnapuram, Robust clustering methods: a unified view, IEEE Trans. Fuzzy Systems 5 (2) (1997) 270–293.

[30] E. Trauwaert, L. Kaufman, P. Rousseeuw, Fuzzy clustering algorithms based on the maximum likelihood principle, Fuzzy Sets and Systems 42 (1991) 213–227.

[31] Y. Hamamoto, Y. Fujimoto, S. Tomita, On the estimation of a covariance matrix in designing parzen classifiers, Pattern Recognition 29 (10) (1996) 1751–1759.

[32] J.V. Ness, On the Dominance of non-parametric Bayes rule discriminant algorithms in high dimensions, Pattern Recognition 12 (1988) 355–368.

[33] F. Kimura, K. Takashima, S. Tsuruoka, Y. Miyake, Modified quadratic discriminant functions and the application to character recognition, IEEE Trans. Pattern Anal. Mach. Intell. 9 (1) (1987) 149–153.

[34] K. Fukunaga, Introduction to Statistical Pattern Recognition, 2nd edition, Academic Press, New York, 1990.

[35] X.L. Xie, G.A. Beni, Validity measure for fuzzy clustering, IEEE Trans. Pattern Anal. Mach. Intell. 13 (8) (1991) 841–846.
[36] R.P. Nikhil, J.C. Bezdek, On cluster validity for the fuzzy C-means model, IEEE Trans. Fuzzy Systems 3 (3) (1995) 370–379.
[37] M. Ramze Rezaee, B.P.F. Lelieveldt, J.H.C. Reiber, A new cluster validity index for the fuzzy C-mean, Pattern Recognition Lett. 19 (1998) 237–246.
[38] N.R. Pal, J.C. Bezdek, On cluster validity for fuzzy c-means model, IEEE Trans. Fuzzy Systems 3 (3) (1995) 370–379.
[39] N.R. Pal, J.C. Bezdek, Correction to on cluster validity for the fuzzy-C-means model, IEEE Trans. Fuzzy Systems 5 (1) (1997) 152–153.
[40] M. Kubat, Decision trees can initialize radial basis function networks, IEEE Trans. Neural Networks 9 (5) (1998) 813–821.
[41] P. Murphy, D. Aha, UCI repository of machine learning databases [machine-readable data respository], Technical Report, University of California, Ivrine, CA. http://www.ics.uci.edu/AL/ML/Machine-Learning.html.

About the Author—FAIZA BEHLOUL received an engineering degree from the Institute of Computer Science, Algiers, Algeria, in 1993 and a Ph.D. degree in Computer Science in 1999 at the National Institute of Applied Science of Lyon, France. She is currently a research associate at the Division of Image Processing, Leiden University Medical Center. Her research interests include soft computing, knowledge driven medical image segmentation and data fusion.

About the Author—BOUDEWIJN P.F. LELIEVELDT received M.Sc. degree in 1994 from the Delft University of Technology, Department of Mechanical Engineering, followed by a Ph.D. degree in 1999 at the Division of Image Processing of the Leiden University Medical Center. Currently, he is employed as a senior research associate, coordinating the development of knowledge driven medical image segmentation methods at the Division of Image Processing. His research interests are in the field of model based object recognition, knowledge driven segmentation and fuzzy logic.

About the Author—ABDEL-OUAHAB BOUDRAA graduated from the Institute of Physics, Constantine University, Algeria, in 1987. He received a University degree in Nuclear Magnetic Resonance in 1993, a Ph.D. degree in Biomedical Engineering in 1994 and University degrees in Statistics and Modeling in 1995 and Positron Emission Tomography in 1997 all from the University of Claude Bernard, Lyon 1, France. He is currently Associate Professor of Electrical Engineering at French Naval Academy, Brest, France. His current research interests include computer vision, vector quantization, data structures and analysis, hard and fuzzy pattern recognition and applications of fuzzy set theory to medical image. Dr. Boudraa is an Associate Member of IEEE Society.

About the Author—JOHAN H.C. REIBER is a professor of Medical Image Processing at the Leiden University Medical Center and the Inter-University Cardiology Institute of the Netherlands (ICIN). He received his M.Sc. degree from the Delft University of Technology in 1971 and the Ph.D. degree in Electrical Engineering in 1975 from Stanford University, California, USA. In 1977 he founded the Laboratory for Clinical and Experimental Image Processing (LKEB) at the Thoraxcenter in Rotterdam, directing the research at the development and validation of objective and automated techniques for the segmentation of cardiovascular images, in particular for quantitative coronary arteriography (QCA), nuclear cardiology and echocardiography. With the move of LKEB in 1990 to the Leiden University Medical Center (LUMC), the scope broadened to intravascular ultrasound, MRI, CT, etc., also in radiological applications. His research interests include (knowledge guided) image processing and its clinical applications. Professor Reiber is a member of the Royal Dutch Academy of Sciences, Fellow of the European Society of Cardiology, senior member of the IEEE and since 1990 he is editor-in-chief of the International Journal of Cardiac Imaging.