Using multilayer perceptrons as receptive fields in the design of neural networks

Mario G.C.A. Cimino [c,*], Witold Pedrycz [a,b], Beatrice Lazzerini [c], Francesco Marcelloni [c]

[a] Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta, Canada T6G 2G7
[b] Systems Research Institute, Polish Academy of Sciences, 01-447 Warsaw, Poland
[c] Dipartimento di Ingegneria dell'Informazione: Elettronica, Informatica, Telecomunicazioni, University of Pisa, Via Diotisalvi 2, 56122 Pisa, Italy

A B S T R A C T

In this paper, we propose a new neural network architecture based on a family of referential multilayer perceptrons (RMLPs) that play a role of generalized receptive fields. In contrast to "standard" radial basis function (RBF) neural networks, the proposed topology of the network offers a considerable level of flexibility as the resulting receptive fields are highly diversified and capable of adjusting themselves to the characteristics of the locally available experimental data. We discuss in detail a design strategy of the novel architecture that fully exploits the modeling capabilities of the contributing RMLPs. The strategy comprises three phases. In the first phase, we form a "blueprint" of the network by employing a specialized version of the commonly encountered fuzzy C-means (FCM) clustering algorithm, namely the conditional (context-based) FCM. In this phase our intent is to generate a collection of information granules (fuzzy sets) in the space of input and output variables, narrowed down to some certain contexts. In the second phase, based upon a global view at the structure, we refine the input–output relationships by engaging a collection of RMLPs where each RMLP is trained by using the subset of data associated with the corresponding context fuzzy set. During training each receptive field focuses on the characteristics of these locally available data and builds a nonlinear mapping in a referential mode. Finally, the connections of the receptive fields are optimized through global minimization of the linear aggregation unit located at the output layer of the overall architecture. We also include a series of numeric experiments involving synthetic and real-world data sets which provide a thorough comparative analysis with standard RBF neural networks.

Keywords:
Conditional clustering
Local modeling
Neural receptive fields
Radial basis function (RBF) networks
Referential neural networks

1. Introduction

In information processing, the common idea of nonlinear neural regression encompasses all data at once and concerns the tuning of all parameters (connections) over the overall training set, which is typically spread across the entire universe of discourse. In general, this learning scheme is usually referred to as global modeling [1]. The explosion of units needed for the realization of more complex mappings leads to increasing difficulties in modeling and, from the practical perspective, brings a danger of inability to cope with highly nonlinear relationships. On the contrary, the concept of receptive fields (RFs) is related to local modeling, i.e., it relates to sub-models that focus predominantly on some selected regions of the entire modeling domain. An overall model is then formed by combining such local models. The most popular examples of architectures based on RFs are the RBF (radial basis function) neural networks [2].

Very likely, RFs are the most prominent and ubiquitous computational mechanism employed by biological information processing systems [3]. For instance, visual, auditory and somatosensory RFs provide local maps that are representations of the corresponding sensory organs, and more complex RFs form when features from several of these cortical maps combine [4,5].

From a regression analysis standpoint, RFs can be considered as collections of units each related to a segment (namely *receptive area*) of the target mapping (which could be conceptual rather than metric) and globally connected to support further levels of processing.

The novelty of the undertaken study relates to a way in which such RFs are being formed and optimized. Instead of simple RBFs used in RBF neural networks, in this paper we consider more complex local RFs, namely referential multilayer perceptrons (RMLPs). The MLP can be considered referential since it "compares" the current input/output with the reference-prototypes of the local receptive area. We show that this architectural

* Corresponding author. Tel.: +39 50 2217 455; fax: +39 50 2217 600.
E-mail addresses: m.cimino@iet.unipi.it (M.G.C.A. Cimino), pedrycz@ece.ualberta.ca (W. Pedrycz), b.lazzerini@iet.unipi.it (B. Lazzerini), f.marcelloni@iet.unipi.it (F. Marcelloni).

development provides an effective way of designing nonlinear mappings by offering an increased level of flexibility due to the different complexities (possible number of neurons) that each RF of this nature could have.

Experiments carried out for a synthetic data set and well-known regression benchmarks show that the proposed architecture achieves high accuracy values while exhibiting limited complexity. Furthermore, we show that in all data sets, our architecture outperforms RBFs.

The paper is organized as follows. In Section 2, we introduce the neural networks based on RFs by describing the RBFs. Section 3 describes the proposed neural network architecture based on RMLPs and in Section 4 we propose a design method for this architecture. In Section 5, we discuss the experimental results. Finally, Section 6 offers several conclusions.

2. RFs networks: a comparative insight

Traditional RFs focus on a "local" character of processing by exploiting a collection of simple units and concentrating all computation around them [6]. As a starting reference in this study, let us consider a close relative of the proposed architecture, namely RBF neural networks, which represent one of basis categories of neural RFs [7]. Here, the topology of each RF is fixed and a number of units, being highly circumstance dependent, are needed to cope with complex mappings. Furthermore, the choice of the form of these RFs is often critical to the successful performance of the networks. A great deal of research has been performed for different types of nonlinear functions so as to replace classical RBFs in RBF architecture [8–13]. As a matter of fact, it can be noticed how the shape of the resulting mapping is determined by the shape of the single RFs. For instance, Gaussian RBF (GRBF) units exhibit a very regular shape, say ellipsoidal or circular, and evidently Gaussian RFs are of hyperellipsoidal shape. The geometry of the field is determined by the class of the RFs used in these networks.

In general, it is not always possible to find the best approximation within a specified class of approximators, even when the analytical expression of the function is given [14].

A central design issue in GRBF-based regression is the selection of an appropriate dimension (number of units) for the representation. Another important design problem is concerned with a suitable choice of parameters of each Gaussian RF. In a standard GRBF network, the training consists of adjusting spreads and centers, as well as weights of the output layer. RBF networks may require a larger number of neurons than standard "global" MLPs and in general they perform best when several training vectors are available.

As it has been proven in [15], if the activation function in the hidden layer is nonlinear, a three (input, hidden and output) layer MLP is a universal approximator. In the MLP networks, where the most frequently used activation function of the hidden layer is the sigmoid, each neuron of the hidden layer can be associated with the overall input domain. Indeed, when a weight between two nodes is modified, for instance by the back-propagation (BP) updating rule during the training phase, the effect involves an infinite region of the input space and can affect large part of the co-domain of the target function. On the contrary, in RBF networks, while similar in terms of topological structure each hidden neuron is associated with a convex closed region in the input domain (*receptive area*), where its response is significantly greater than zero, and dominates over every other neuron. Changing the size of the region of the input space, in which the activation function of a neuron in an RBF RF fires, or shifting its position produce an effect *local* to the region dominated by that neuron. In general, this locality property of the RBF allows the network layout to be incrementally constructed adjusting the existing neurons and/or adding new ones. Indeed, since any parameter tuning that involves an RBF RF has a local effect, the knowledge encoded in the other parts of the network is not lost; hence, it is not necessary to go through a global revision process. Furthermore, there is the possibility of providing a logic interpretation of the hidden neuron semantics, as the closed regions corresponding to neuron activation areas can be labeled and interpreted as elementary concepts, to develop a logic-based model of neurocomputing, cf. [16,17].

The knowledge needed to distribute RFs in the input space can come from a designer-oriented data analysis or from some preliminary analysis of available data.

In the first case, the RFs are formed based upon some qualitative hints provided by the designer who distributes the RFs in the input space according to her/his domain knowledge. This allows focusing attention of the training mechanisms on some essential regions of the input space. Such an orientation could eventually lead to a multi-resolution (accuracy) character of processing carried out by the neural network and accelerate the learning itself. Along this line, some fuzzy systems emerge. In particular, some interesting analogies between RBF neural networks and rule-based systems have been already discussed in [18]. Here, the antecedents of the respective rules describing the fuzzy model give rise to a so-called linguistic partition of the input space. Depending on whether the output vector is involved in the clustering process or not, there can be an input–output or an input clustering [19,20]. Typically, the application domain is too complex and the designer can provide no qualitative hint to distribute RFs. Thus, the second approach to distribute RFs in the input space has to be considered.

The original data set is analyzed with respect to its internal topology by typically using clustering methods [19]. For instance, in RBF networks, clustering techniques associate a cluster with each RBF node in the hidden layer. In [21] the importance of the initialization in the clustering algorithm for RBF networks was studied, and then a new algorithm was proposed. Of course, the results of the clustering algorithm directly affect the performance of the RBF networks. In [22,23], performance achieved by different clustering techniques, namely K-means [24], iterative optimization (IO) [25], a technique based on depth first (DF) search [26], two combinations of the previous techniques [27] (in the first combination, named IODF, IO is applied to the odd iterations and DF to the even iterations; in the second, named DFIO, DF is applied to the odd iterations and IO to the even iterations), on-line K-means [28] and optimal adaptive K-means [29], are compared. These results indicate that no technique was clearly better than the others. The best overall performances for the training, validation and test subsets were obtained by the optimal adaptive K-means (500 clusters), DF (400 clusters) and DFIO (400 clusters), respectively.

The evident drawback that is commonly encountered across the variety of the clustering methods utilized to determine the distribution of RFs is that all of them are completely unsupervised [2]. Actually, for this task, it would be extremely advantageous to establish groups within the data ensuring that these are also homogeneous with respect to the output variable. Conditional clustering has proved to be an effective approach to achieve this property of homogeneity [30]. In conditional clustering the process of revealing the structure in the input space is conditioned upon some linguistic landmarks defined in the output space. These landmarks correspond to classes in classification problems and to fuzzy sets which partition the output variable in regression problems. Partitions of the output variable can be defined by the user or, as in our approach, generated by applying a clustering

algorithm to the output values. In conclusion, a limitation of the use of simple RFs is that they tend to be memory intensive.

A different approach based on the aggregation of more complex sub-models is represented by the broad class of modular neural networks (MNs) [31–33]. The use of MNs allows capturing the structure of an input–output mapping at intermediate levels of granularity, in order to permit the formation of higher-order computational units that can perform more complex, explicit and interpretable tasks. From a technical perspective, although the network presented in the next Section is not strictly based on MLP, it could be treated also as a novel structure of MNs.

3. Architecture of the network

The proposed network topology based on RFs, as visualized in Fig. 1 in the case of two-dimensional input space and mono-dimensional output space, consists of two types of functional blocks arranged in a two-layer architecture. The first functional layer is formed by a collection of RFs (sub-models RF_1,RF_2,…,RF_N), whereas the second layer is a single unit (referred here as an *AGGREGATOR*) whose role is to aggregate the values coming from the RFs.

Each RF comes with its own prototype (shown as a black dot) and its corresponding receptive area (gray region), in both input and output spaces. A receptive area allows a RF to concentrate (focus) in a certain sub-region of the overall domain, thus realizing a computing of local nature.

Let us suppose that the receptive areas in the input space are overlapped, as well as the corresponding segments in the output space, and that any input $\mathbf{x}$ is included in two receptive areas, i.e., there are two RFs that contribute to compute the output value y corresponding to $\mathbf{x}$. In the following, let us denote as "context" each sub-region of the output domain.

The extraction of a context for each RF leads to split the problem into sub-problems. This is done by a procedure guided by the context constraints, i.e. fuzzy sets or fuzzy relations on the output space (*context variable*). Furthermore, this process supplies a set of prototypes, i.e. representatives in the input space. As mentioned before, the computing realized at the level of the individual RF is also *referential*, that is, the current input/output is compared with the reference-prototypes of the local sub-region. Finally, RFs are connected through the further processing phase, the aggregation [17].

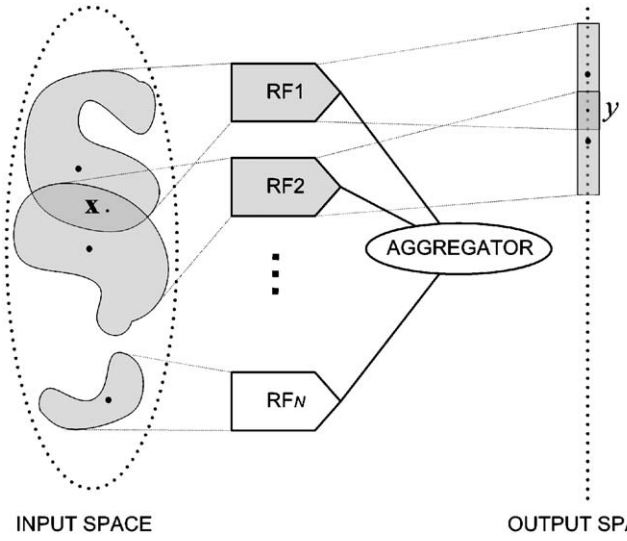

Fig. 1. A topology that employs generalized receptive fields.

In Fig. 2, we show in more detail the architecture in Fig. 1, by highlighting the implementation of each component. The MLPs, that build a nonlinear mapping from $\Re^F$ to $\Re$ and serve as core of the generalized and highly heterogeneous RFs, are organized in P groups, one for each context. The i-th group contains C_i MLPs, and the j-th MLP holds the reference point $\mathbf{r}_{i,j} \in \Re^F$, determined in the input space for context i. For a given input $\mathbf{x}$, for each group i, module $A(\cdot)$ performs a selection of the MLPs belonging to that group by enabling to feed only the $MLP_{i,\bar{j}}$ whose corresponding reference point $\mathbf{r}_{i,\bar{j}}$ is the closest to the input $\mathbf{x}$ among all the reference points $\mathbf{r}_{i,j}$. $MLP_{i,\bar{j}}$ receives the difference between the current input $\mathbf{x}$ and the corresponding $\mathbf{r}_{i,\bar{j}}$ as input. The output of $MLP_{i,\bar{j}}$ is shifted by the corresponding reference point $v_i \in \Re$, determined in the output space, and then weighted by a certain activation function $\phi_i(\cdot)$.

Each MLP is aimed at the mapping of an input receptive area, whose prototype is the reference point $\mathbf{r}_{i,j}$, to a specific output area, whose prototype is the reference point v_i. Thus, the term RMLP has been employed to denote an MLP inclusive of its own references.

The role of $\phi_i(\cdot)$ is to activate (weight) the output of the neural network, depending on the reference v_i. The weighted outputs are then processed by module $\Omega(\cdot)$, which acts as a filter by selecting a pair of consecutive weighted outputs, on the basis of a specified criterion. Finally, the linear node in the output layer produces a linear combination of the selected outputs, with weights w_i.

More specifically, the output value of the activation function $\phi_i(\cdot)$ is defined as its input value multiplied by the contextualized value of this input. Formally speaking, $\phi_i(\cdot) \equiv (\cdot) \cdot F_i(\cdot)$, where "$(\cdot)$" stands for the identity function and $F_i(\cdot)$ for the "context" function, respectively. Using the context function, the output of the RMLP narrows down to the range of values within the field where it has been trained. Outside this range, the output of the field is set to zero, because its processed value is not meaningful.

Actually, contexts overlap in the output variable, and for each input $\mathbf{x}$ to the system, only the outputs of two adjacent fields are supposed to be meaningful (these fields are exactly selected by

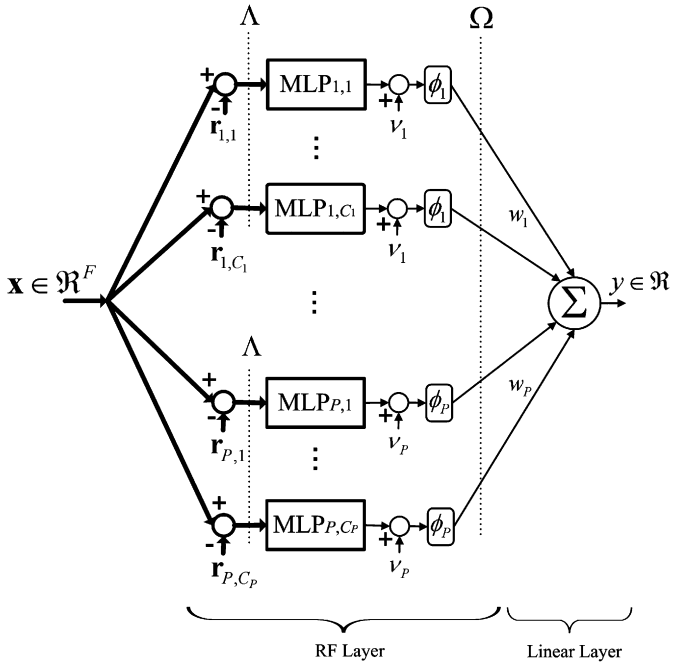

Fig. 2. An overall architecture of the network.

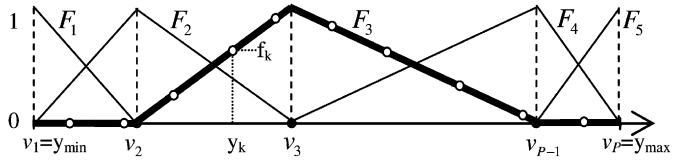

Fig. 3. Triangular weighting functions (membership functions) induced on the output variable (here $P = 5$).

$\Omega(\cdot)$, as described below). Therefore, the context functions split the output variable into overlapped bands as in Fig. 3, associating with each output reference v_i the triangular function $F_i(\cdot)$, whose core and support are $\{v_i\}$ and $[v_{i-1}, v_{i+1}]$, respectively. Similarly, the minimum and maximum of the output values are considered as the modal values of corresponding triangular functions $F_i(\cdot)$ and $F_P(\cdot)$, respectively. Here P is the number of contexts. We can regard these contexts as a certain vehicle to decompose the problem, by considering the output space (context variable) in which a fuzzy condition can be derived, or expressed by terms such as "low", "medium", "high", etc.

It is worth stressing that contexts could originate from different sources. For instance, these fuzzy sets could come from the user/designer who is interested in some particular relationships, say "determine the sub-model M under the condition that the output is linguistically expressed as some fuzzy set D". There could be application-oriented needs which may call for models that are focused on some specific requirement implied by the problem itself. Say, we may be required to develop a model which is oriented on *high* positive output values of the process. Interestingly, in this way the context being imposed can help handle issues of imbalanced (rare) data which we would be interested in careful modeling. Note also that contexts are ordered on the basis of references, and then there are pairs of *adjacent* contexts, whose supplied values are combined to produce the output of the system.

More specifically, let us now consider the selection module $\Omega(\cdot)$ as shown in Fig. 2. It selects, among the P values associated with the contexts, the two values which correspond to adjacent contexts and are the closest to each other, setting the remaining $P-2$ values to zero. To explain the rationale behind this module, consider again the overall information flow in the architecture. For a given input $\mathbf{x}$, due to the local selectors, P RMLPs, one for each context, supply a value in output, and these values are shaped using the activation functions $\phi_i(\cdot)$. However, only the output values of two RMLPs should be considered, i.e., the RMLPs whose contexts *contain* that input. For instance, for the input corresponding to y_k in Fig. 3, only the two RMLPs corresponding to F_2 and F_3 should be considered. Each RMLP is trained by using only the training patterns whose outputs belong to the context of the RMLP and whose inputs are selected by $\Lambda(\cdot)$. Thus, a given input is *proper* only for two RFs, and *improper* for the other $P-2$. RFs are trained to recognize just their proper inputs. However, for a given improper input, an RF may supply a misleading output. For instance, it may supply the $\Omega(\cdot)$ module with a value different from zero, because the output value of the field may by chance occur where its context function is not zero. Module $\Omega(\cdot)$ is, therefore, needed to select only the two adjacent RFs for which the inputs are proper, i.e., the two RFs that were trained to recognize that input. This selection can be performed considering that these RFs provide approximately the same output values for the same proper inputs. Though there is still a minimal chance that another improper RF may supply an output value similar to the one supplied by a proper RF, we observed that this chance is not relevant in practice.

In conclusion, the output of the overall system can be formally expressed as

$$\mathrm{out}_i(\mathbf{x}) = \mathrm{MLP}_{i,\hat{j}}(\mathbf{x} - \mathbf{r}_{i,\hat{j}}) + v_i;$$

$$y(\mathbf{x}) = \sum_{i=1}^{P} \Omega[\mathrm{out}_i(\mathbf{x}) \cdot F_i(\mathrm{out}_i(\mathbf{x}))] \cdot w_i; \quad (1)$$

with $\hat{j}$ such that $\|\mathbf{x} - \mathbf{r}_{i,\hat{j}}\|$ is minimum, $1 \leqslant j \leqslant C_i$.

It can be noticed, in Fig. 2, that the proposed architecture is divided in two layers that are related to corresponding blocks in the pattern of Fig. 1. In particular, the linear layer plays the role of aggregator and the i-th RMLP, in conjunction with the activation function $\phi_i(\cdot)$, plays the role of RF.

As it is evident from the description of the architecture, it comes with a high level of flexibility which could be fully exploited in its development.

4. The development of the network

As already stated, the development of the network comprises several major phases. We start from building a skeleton (blueprint) of the network and afterwards refine its parameters through detailed parametric learning. The blueprint is formed by using a conditional (context)-based clustering [17]. The structure formed in this manner is optimized by training the individual MLP neural networks and adjusting the weights of the output linear aggregator. In the following, we elaborate on the details of the construction by delineating the successive design steps.

Let $y = f(\mathbf{x})$, with $\mathbf{x} \in \Re^F$, be the unknown mapping to be identified by using a set T of N input–output pairs $(\mathbf{x}_k, y_k)$, with $y_k = f(\mathbf{x}_k)$. Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}, \mathbf{x}_k \in \Re^F$, and $Y = \{y_1, \ldots, y_N\}, y_k \in \Re$ be, respectively, the set of the inputs and the set of corresponding outputs of the patterns contained in T.

4.1. Blueprint development

The blueprint development is realized as follows:

1. Find the representatives in the output space by using a standard k-means clustering algorithm applied to the set Y of training outputs. Let $P-2$ be the number of clusters, and $V = \{v_2, \ldots, v_{P-1}\}, v_i \in \Re$, the set of corresponding prototypes.
2. Assume that $v_1 = \min(Y)$ and $v_P = \max(Y)$. Partition the output domain y with P triangular membership functions $F_i(y)$, denoted as context fuzzy sets, in the following way. The core and support of $F_i(y)$, $i = 1 \ldots P$, are $\{v_i\}$ and $[v_{i-1}, v_{i+1}]$, respectively, with $v_0 = y_{\min}$ and $v_{P+1} = y_{\max}$.
3. For each context fuzzy set $F_i(y)$, apply the conditional fuzzy C-means (CFCM) [17] to $\mathbf{X}$, subject to the context $F_i(y)$, so as to derive a fuzzy partition of C_i clusters. Let $\mathbf{r}_{i,j} \in \Re^F$, $1 \leqslant j \leqslant C_i$, be the prototypes of these clusters. The value C_i might be prefixed, or generated using a cluster validity index such as the one introduced by Xie and Beni [34].

4.2. The parametric learning of the RFs

Let $\mathbf{X}_{i,j}$ be the subset of $\mathbf{X}$ which contains the input values that belong to the cluster represented by $\mathbf{r}_{i,j}$ with the highest membership degree among all C_i clusters. Use the pairs $(\mathbf{x}_k - \mathbf{r}_{i,j}, y_k - v_i)$, where $\mathbf{x}_k \in \mathbf{X}_{i,j}$ and y_k is the output corresponding to $\mathbf{x}_k$, to train the MLP associated with $\mathbf{X}_{i,j}$. The gradient descend method used in the BP procedure adopts the following

cost function (performance index):

$$MSE_{i,j} = \frac{1}{|\mathbf{X}_{i,j}|} \cdot \sum_{\substack{\mathbf{x}_k \in \mathbf{X}_{i,j}, \\ y_k \in \mathbf{Y}_{i,j}}} [MLP_{i,j}(\mathbf{x}_k - \mathbf{r}_{i,j}) - (y_k - v_i)]^2, \tag{2}$$

where "$|\cdot|$" denotes the cardinality function. Note that the network is trained using referenced target values, namely $y_k - v_i$. Note also that the approximation of each $MLP_{i,j}$ concerns solely the subset $\mathbf{X}_{i,j}$ of $\mathbf{X}$.

4.3. The optimization of the linear neuron

Once the MLPs have been trained, the connections of the linear combination, i.e., the parameters (weights) $\boldsymbol{w} = \{w_1,\ldots,w_P\}$ are determined through the standard LSE optimization. More formally, the linear system of N equations in P variables is determined by choosing $\boldsymbol{w}$ that $Q = \sum_{k=1}^{N}(\tilde{y}_k - y_k)^2$ attains its minimal value, where $\tilde{y}_k = \sum_{r=1}^{P} \Omega[\mathrm{out}_i(\mathbf{x}_k) \cdot F_i(\mathrm{out}_i(\mathbf{x}_k))] \cdot w_i = \sum_{i=1}^{P} \sigma_{ik} w_i$. This minimization problem can be expressed as $\min Q(w_1,\ldots,w_P) = \min \sum_{k=1}^{N} (\sum_{i=1}^{P} \sigma_{ik} w_i - y_k)^2$, whose solution is

$$\mathbf{w} = (\boldsymbol{\Sigma}^{\mathrm{T}}\boldsymbol{\Sigma})^{-1}\boldsymbol{\Sigma}^{\mathrm{T}}\mathbf{y}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{11} & \ldots & \sigma_{1N} \\ \ldots & \ldots & \ldots \\ \sigma_{P1} & \ldots & \sigma_{PN} \end{bmatrix},$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \ldots \\ y_N \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} w_1 \\ \ldots \\ w_P \end{bmatrix}.$$

4.4. Further considerations on the structuring of the RMLP

Let us consider the dimensioning of the local RFs, which are realized in terms of standard fully connected MLPs. In general, structuring a single "centralized" MLP architecture requires identifying the correct number of hidden nodes and the number of layers [35,36]. Trial-and-error is the most practical method for determining this value even if it is very time expensive. Furthermore, the common BP technique is a local search algorithm and thus tends to become trapped in local optima. To solve this problem, a variety of approaches have been used [36], which, however, require some domain knowledge to tackle the necessary increase in the training complexity. Actually, as the neural architecture becomes less complex, which is the case of MLPs used as RFs in our approach, the BP algorithm is more likely to be successful, and the most related issues practically disappear. This corroborates the local modeling paradigm as fundamental approach to cope with model complexity. In particular, good generalization properties can be obtained using simple and well-known standard rules-of-thumb for MLP structuring and training, and employing the same criteria for all data sets. This means that no domain knowledge is injected in RF architecture through specific tuning processes. Then, a comparison with the RBF architecture can be fair since in both systems no user parameter is tuned on the training set, except the network size (i.e., the number of units), which is scanned in both architectures.

A number of studies in the literature [37–39] indicate the superiority of two hidden layered neural networks over one hidden layered networks, with a finite number of hidden units, in terms of mapping capabilities. In particular, it is useful to have several units in the second hidden layer because they enable the net to fit local critical points, and an MLP with two hidden layers can often yield an accurate approximation with fewer connections (weights) than an MLP with one hidden layer. In fact, it has been experimentally verified that the same accuracy can be obtained with RFs with both one and two hidden layers, but using, in case of one hidden layer architecture, a number of parameters 4–5 times greater than in two hidden layer architecture. Further, one of the parameters that influence the generalization capability of networks is the complexity, closely related to number of weights in the architecture [40]. Therefore, to ensure the use of the minimum number of hidden units, it is useful to start from training with a very small network, and growing the network by adding a hidden unit until the gradient descent fails to find a satisfactory solution. Not surprisingly, this process is very fast and domain independent, due to the low complexity of RFs.

To fully evaluate the success of the proposed neural network architecture with respect to RBF networks, we consider both accuracy and complexity. Network complexity is commonly measured by the Vapnik–Chervonenkis (VC) dimension [41]. Baum and Haussler [40] have shown that the VC dimension is closely related to the number of weights (parameters) in the architecture, and since the exact VC dimension is practically impossible to calculate for complex networks, the number of parameters (tunable real numbers) is used as an approximate indicator of complexity in what follows.

Considering a fully connected MLP with N_1 and N_2 units in the first and second hidden layer, respectively, as well as F (i.e. the input space dimension) units in the input layer, and one (i.e. the output space dimension) unit in the output layer, the total number of parameters employed per MLP is

$$W_{MLP} = F \cdot N_1 + (N_1 + 1) \cdot N_2. \tag{3}$$

In addition, for the proposed architecture (see Fig. 2), each RF needs an input reference (i.e., F parameters). Furthermore, each context needs an output reference (i.e., one parameter) and a context triangular function (i.e. three parameters). Actually, the triangular function is fully determined through the output references, and thus it is not considered in parameter computation. Finally, the architecture employs a parameter in the linear layer, per context. In conclusion, the number of parameters needed per context is

$$W_{CONTEXT_i} = (N_1 + 1) \cdot (N_2 + F) \cdot C_i + 2. \tag{4}$$

Considering $C_1,\ldots,C_P$ (number of MLPs per context), the total number of parameters becomes

$$W_{MLP\text{-}SYSTEM} = 2P + (N_1 + 1) \cdot (N_2 + F) \cdot \sum_{i=1}^{P} C_i. \tag{5}$$

Note that, as the output dimension is just one, the contribution of N_1 in terms of number of parameters, i.e. $N_1 \cdot N_2 + N_1 \cdot F$, is more relevant than the one of N_2, i.e. $N_1 \cdot N_2 + N_2$.

As regards the comparative architecture, in the learning scheme, each RBF uses F parameters, i.e. the center coordinates in the input space (as explained in the next section, we fix the spread). In addition, the architecture employs a further parameter in the linear layer. Finally, the number of parameters needed per RBF is

$$W_{RBF} = F + 1 \tag{6}$$

and for the overall RBF system

$$W_{RBF\text{-}SYSTEM} = (F + 1) \cdot P, \tag{7}$$

respectively.

To show the effectiveness of the proposed system, giving also a comparison with the RBF architecture, in the next section we consider first an illustrative example and then a quantitative analysis on a number of well-known data sets.

5. Experimental studies

In this section, we report on a comprehensive suite of numeric experiments. Our intent is to demonstrate the performance of the network and contrast its capabilities with commonly encountered RBF neural networks. Both some synthetic data and real-world data are considered. In all experiments, the data are normalized (by subtracting the mean value and dividing the data by their standard deviation).

As regards RBF networks (RBFNs), two different learning procedures are dominant in the literature. The *static* learning procedure fixes a number of basis functions and learns by modifying the parameters of the basis functions [42]. On the other hand, the *dynamic* learning procedure also modifies the number of basis functions, integrating the initialization and the refining phases in an incremental learning algorithm [43,44].

A static learning algorithm is parametric because the search for the optimal approximator corresponds to a search in the parameter space defined by the fixed number of RBFs. In absence of domain knowledge, due to the high number of units needed, training process for determining centers and spreads could be ineffective. On the contrary, a dynamic learning algorithm adaptively changes the parameter space in which it operates, by adding (or deleting) RBFs, thus requiring a less precise domain knowledge than a static learning algorithm. The dynamic procedure used in the experiments determines incrementally the number of radial units, iteratively adding a unit at each step. The procedure starts with no neuron in the hidden layer. Hence, the following steps are repeated until the SSE falls beneath an error goal or a prefixed maximum number of units is reached (in the limit, this might be one node per training pattern): (i) the network is simulated; (ii) the input vector with the greatest error is found; (iii) a RF is added with center equal to that vector; and (iii) the output layer weights are recomputed, by solving a set of linear equations, so as to minimize SSE. The dynamic method used in the experiments is described in [44]. In particular, we used an optimized implementation of the method (2005), which has been included in *The MathWorks Matlab*^{TM} v. 7 toolbox.

In the comparative (RBF-based) system, the spreads are fixed to the (standard) value 1.0, as data are normalized. Obviously, it could be helpful to use different spreads of the RFs. This, however, requires further learning where knowledge injection is needed, rather than a universal value of spreads. Also, both architectures could have been better adjusted by exploiting domain knowledge or proceeding with more advanced learning. However, the experimentation has been carried out avoiding complex solutions for both systems so as to reduce the possibility of customization, i.e., the injection of some form of domain knowledge.

As regards MLPs used in the proposed system, we consider $N_1 = 5$ and $N_2 = 3$ as maximum number of neurons being located in the first and second hidden layers, respectively. These values result from some preliminary experimentation, in which we have observed that the performance of the MLP system is quite independent of the size of the network and the distribution of training data. Hence, in function (3), the maximum number of parameters per RF is $W_{MLP} = 5F + 18$. In the training of the networks, we used a predefined number $E = 2000$ of learning epochs. We verified that these epochs are sufficient to arrive at the point where no further relevant improvement of the performance of the network is observable. The learning rate was equal to 0.05 which, albeit quite small, was helpful in assuring stable learning and avoiding possible oscillations.

In the RBF architecture, complexity depends only on the number of radial functions. In the MLP architecture, complexity depends on three parameters: number of neurons employed in the MLPs, number of contexts and numbers of fields per context.

In order to obtain in both systems a similar number of parameters, in the MLP system we have varied the number of contexts, starting from the minimum possible value, i.e. three. Also, for each context, we have determined the number of fields by the Xie-Beni index. In general, other validity indexes could be successfully employed [34]. However, the objective of this experimental section is to demonstrate that, using standard and common components, the MLP architecture outperforms the RBF architecture, when the complexity is comparable. To have the direct control of the number of parameters employed in the MLP architecture, the user could fix a number of fields for each context, thus avoiding the use of a validity index. However, we experienced that the use of the Xie-Beni index is very helpful for large and highly dimensional data sets. Though this index could breed sub-optimal solutions, this is not relevant for the aims of this study.

To compare the systems in the case of very low complexity, in some trials we have also reduced the number of neurons for MLPs (the same number for all MLPs in the system, for the sake of simplicity) until $N_1 = 2$ and $N_2 = 2$. In any trial, the maximum complexity for MLPs is, however, determined by the upper limit of $N_1 = 5$ and $N_2 = 3$ mentioned above.

5.1. Synthetic data set

Let us consider a three-dimensional (i.e., the dimensionality F of input space is equal to 2) synthetic data set consisting of 10,000 elements obtained by sampling a gray level image. Fig. 4a and b shows the image and the data set, respectively. To perform an early qualitative analysis with this pilot example, let us examine a representative training scenario, using a common *holdout* method. The training set, therefore, consists of two-third (66.66%) of the available data, randomly extracted, while the remaining data (33.34%) form the test set.

We applied the above-mentioned dynamic learning procedure to determine the RBF network. The resulting network is composed of 113 units, which correspond to 339 parameters.

Fig. 5 shows the output of the RBF system. The values of the mean square error (MSE) obtained on the training and test sets are 0.0826 and 0.0850, respectively.

As regards the MLP architecture, to guarantee interpretability and keep the number of parameters low, we employed four contexts. Fig. 6 shows the output of our MLP system, using the same training and test sets as for the RBF network. Also, the total number of parameters is almost the same (338): just 11 RFs have been used, with each RF having 30 parameters. Carrying out 20 trials under the same conditions, the MLP architecture achieved an MSE (expressed as average $\pm$ standard deviation) on the training and test sets equals to 0.0384 ± 0.0038 and 0.0415 ± 0.0043, respectively.

We observe that though using a similar number of parameters, the MSE of our system is considerably lower than the MSE of the RBF network. These results are mainly due to the different modeling capabilities of the two types of neural networks. As a matter of fact, RFs implemented by MLPs are able to model more complex shapes than RBFs.

Fig. 7 shows the four context fuzzy sets and the relative frequency distribution of the output points, related to the same trial of Fig. 4, using bold and normal lines, respectively. More specifically, the abscissa value of the k-th point of the function is y_k, whereas the corresponding ordinate value is the number of times that y_k occurs in the training set. The ordinate values are normalized, i.e., divided by the maximum value. Note that just

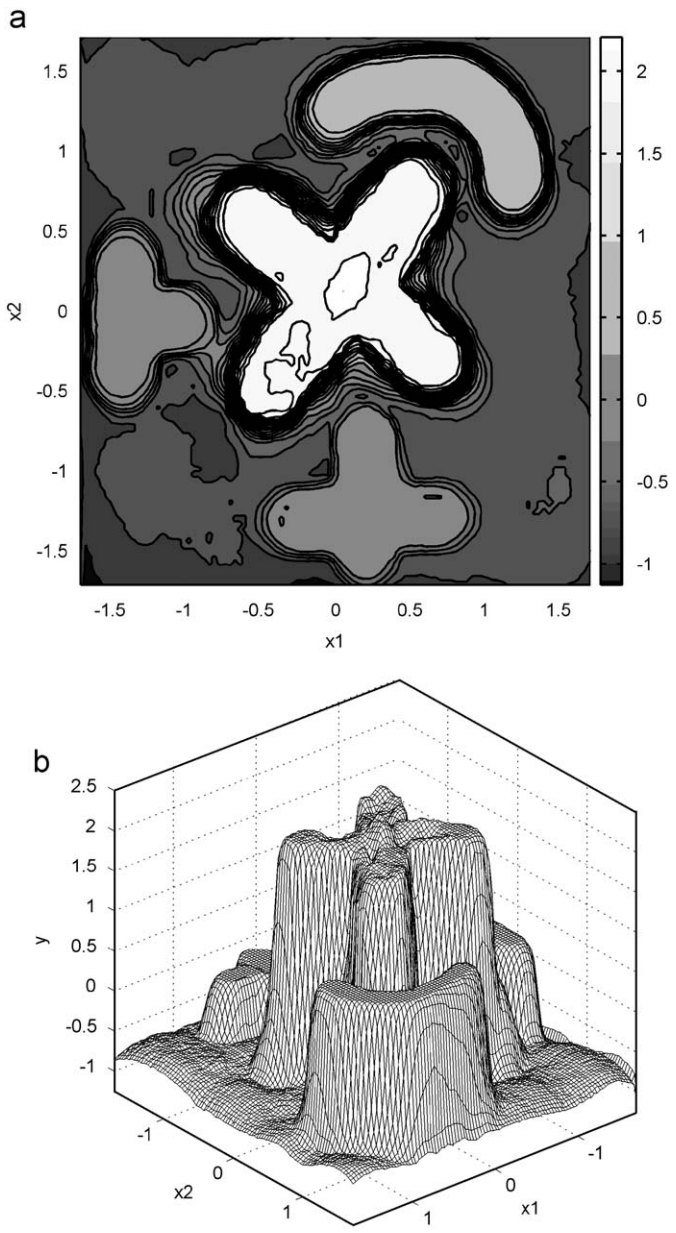

Fig. 4. The *synthetic* data set (b), sampled from a gray level image (a).

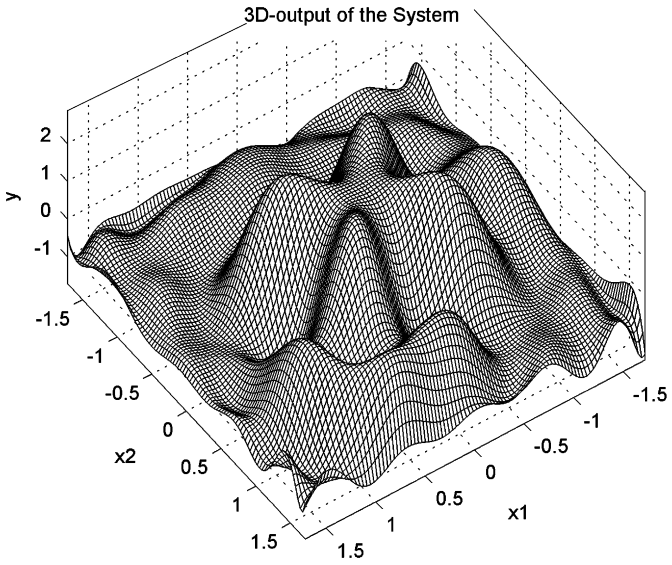

Fig. 5. Output of the RBF system.

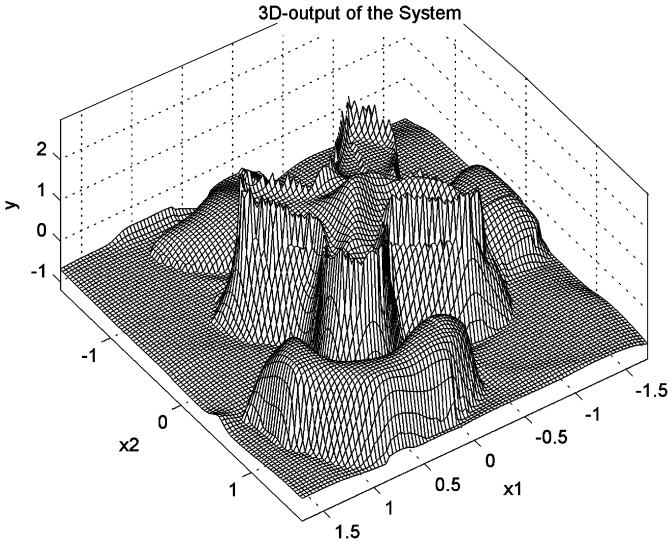

Fig. 6. Output of the MLP system.

four clusters in the output space are not enough to adequately fit the data structure.

Fig. 8 shows clusters and reference points in the input space for each context. In particular, each cluster is represented by a different texture and each reference point by a numbered rectangle. Note that there is a different number of RFs per context, determined by using Xie-Beni index. The figure clearly shows the local nature of each sub-model based on an RF. As the complexity of each RF is kept low, too many RFs do not allow the system to reflect a detailed mapping. At the same time, it becomes obvious that one has to keep the number of parameters much lower than the number of points in the data set. Thus, too many RFs do not allow the system to realize an effective generalization and an efficient training process.

In the next section, we consider a series of numeric experiments involving a number of data sets, in order to provide a thorough comparative analysis. Since a single evaluation is dependent on the data points, which end up in the training and test sets, we adopted a 10-fold cross-validation. For each trial, the training and test sets consist, respectively, of randomly extracted 90% and 10% of the original data.

The comparison between RBF and MLP architecture is made using the same fold cross sections, for increasing number of RFs (parameters). The analysis has been performed up to a reasonable number of parameters, on the basis of the following well-known rule-of-thumb for machine learning systems: the ratio between number of parameters employed in the system and number of observations should be considerably lower than the desired MSE.

5.2. Real-world data

We consider a suite of real data coming from publicly available and well-documented web sites (Bilkent, http://funapp.cs.bilkent.edu.tr/DataSets/; UCI, http://www.ics.uci.edu/~mlearn/MLSummary.html). Table 1 characterizes these data in terms of the dimensionality of the problems as well as the size of the data sets (the first line in the table describes the synthetic data set

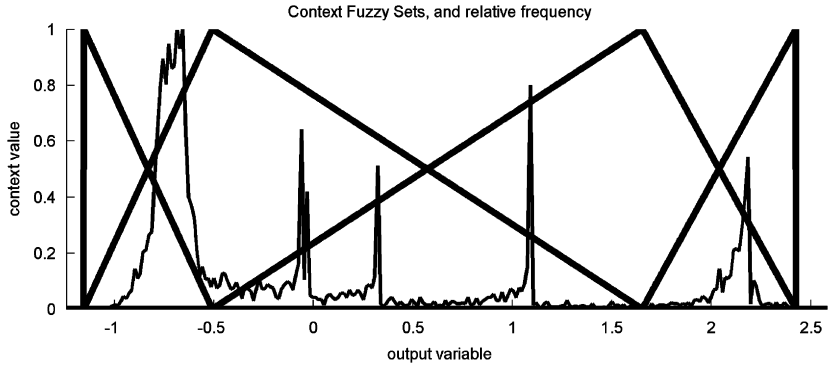

Fig. 7. Context fuzzy sets and relative frequency of the output points.

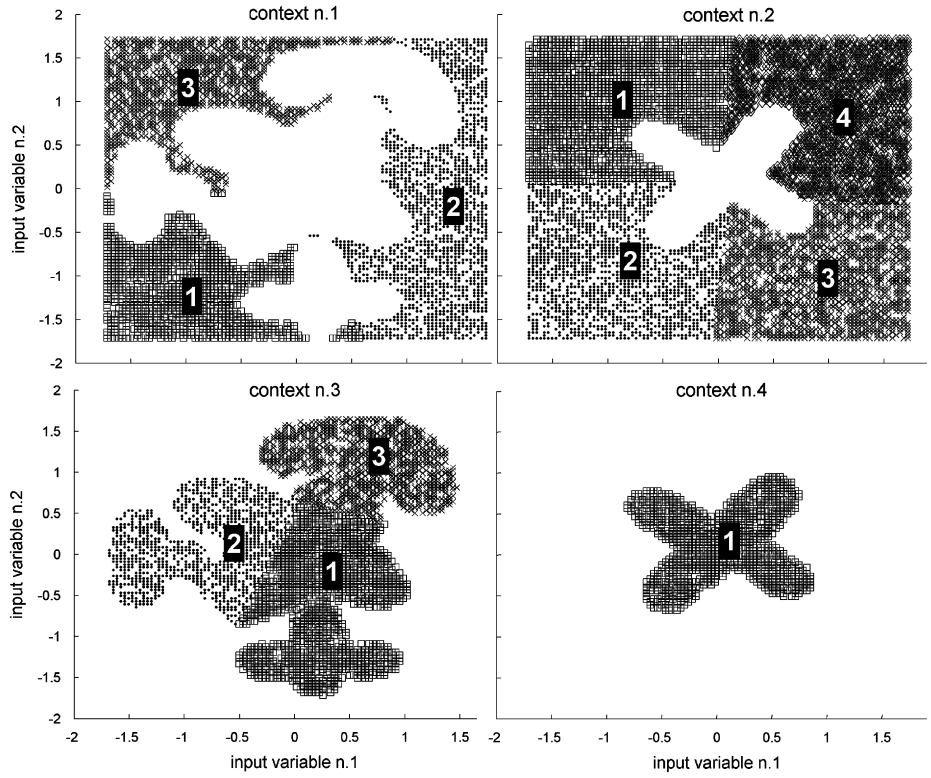

Fig. 8. Clusters and reference points in the input space for each context.

Table 1
Data sets dimensions and max receptive field complexity.

Data set name	Dimensionality of input space (F)	Number of data (N)	Max number of parameters per MLP unit (W_{MLP})	Number of parameters per RBF unit (W_{RBF})
Synthetic	2	10,000	30	3
Abalone	6	4177	54	7
Ailerons	38	7154	246	39
Kinematics	8	8192	66	9
Computer activity	21	8192	144	22
House_16H	15	11,392	108	16

discussed in Section 5.1). Our intent is to consider problems of higher dimensionality than those typically used in other studies. Furthermore, in order to provide a clear insight into the dimensionality of the MLPs and the RBFs, we report the maximum number of the parameters of a single MLP unit and the number of parameters of a single RBF unit.

Note that, to solve scalability problems (which are detailed below in this Section) in the system routine used to generate the

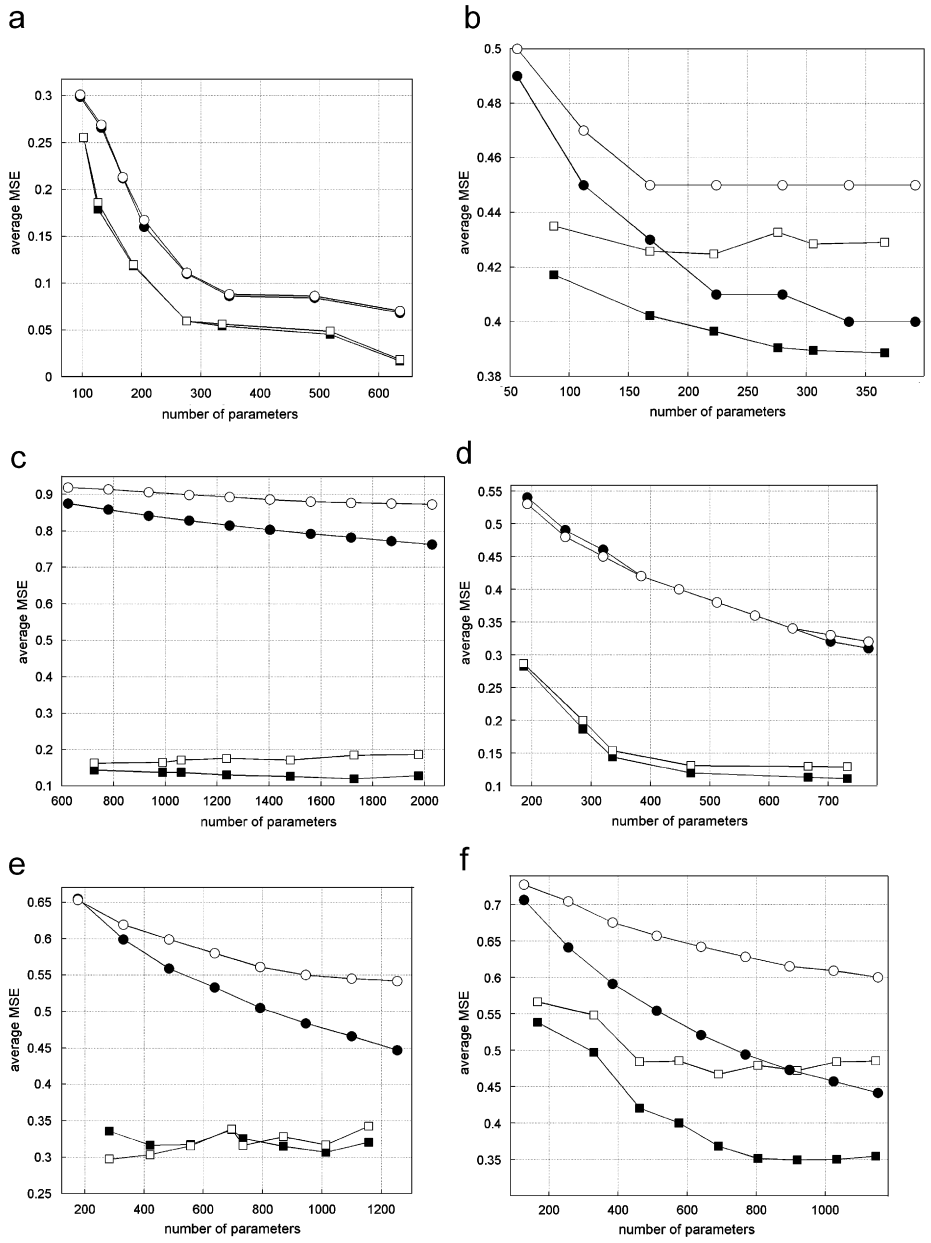

Fig. 9. Average MSE versus number of parameters, for RBF systems on training (●) and testing sets (○), and for MLP systems on training (■) and testing sets (□): (a) synthetic data set; (b) Abalone data set; (c) Ailerons data set; (d) kinematics data set; (e) computer activity data set; and (f) House_16H data set.

RBF architecture, we used only 50% (randomly extracted) of the original data for the *House_16H* data set.

The performances of the resulting architectures on these data sets are represented in Fig. 9. For each data set, the resulting values of the average MSEs versus the number of parameters are jointly plotted for both MLP (squares in the figure) and RBF (circles in the figure) architectures, thus providing a useful comparative insight. For a given number of parameters, the average MSE is evaluated for both the training (black) and test (white) sets.

The networks based on MLPs as RFs demonstrate an evident superiority when compared with the RBF neural networks; this superiority manifests itself both in training and testing phases thus highlighting the generalization properties of our approach. Obviously, both architectures could be improved by exploiting domain knowledge or using a more accurate learning. For example, in the RBF network it could be helpful to use different spreads of the RFs. This, however, would have required further learning where knowledge injection is needed.

Let us examine more precisely the series of data sets used, starting from the synthetic data set (Fig. 9a). Although Gaussian functions should excellently fit smoothed shapes generated by sampling synthetic functions, the MLP RFs outperform the RBF-based architecture. In the case in point, both systems reveal no overfitting problem, having almost the same error on training and test sets. The lowest average MSE (0.0182 using 636 parameters) obtained by the proposed system on test set is almost four times lower than the lowest average MSE (0.07 with exactly the same number of parameters) obtained by the RBF network.

In Abalone data set (Fig. 9b) the nominal attribute has been removed, as usual for non-metric variables. Both systems reveal some difficulties in improving their accuracy and their generalization capabilities, for increasing numbers of parameters. This could be ascribed to some overfitting problem. Despite this, the

proposed architecture shows better generalization capabilities, and further the lowest average MSE (i.e. 0.435, with 87 parameters) is lower than the lowest average MSE of the comparative RBF architecture (i.e. 0.45, with 168 parameters).

On Ailerons data set (Fig. 9c), where two attributes have been removed because they are constant, the difference between the lowest average MSEs of the two approaches is evident, 0.162 for the MLP architecture versus 0.873 for the RBF architecture.

On kinematics data set (Fig. 9d), different modeling capabilities of the two approaches are even more pronounced: the lowest average MSEs on test set are 0.129 for the MLP architecture with 732 parameters and 0.32 with 768 parameters for the RBF architecture, respectively.

On some data sets, as Abalone and computer activity data sets (Fig. 9e), the trend of the MLP system shows some ripple. This can be ascribed to the variance of the learning process that, however, has been verified to be negligible with respect to the differences between the performance levels of both architectures. In the latter data set, the lowest average MSEs are 0.297 (with 282 parameters) and 0.542 (with 1254 parameters), for the MLP and the RBF architectures, respectively, confirming the trend on both accuracy and generalization characteristics. Lastly, on House_16H data set (Fig. 9) the MLP architecture (with the lowest average MSE of 0.467 using 690 parameters) outperforms the comparative architecture (with the lowest average MSE of 0.6 with 1152 parameters).

Furthermore, the MLP learning algorithm is more scalable than the RBF. Indeed, the receptive areas are constructed using a clustering process that concerns just a mono-dimensional space (the output space), and a modularized (context-based) clustering that involves a local computation (input space). Also, each MLP is trained on a subset of the training data, with bounded complexity, thanks to the local nature of the process. Thus, each MLP is trained in a comparatively short computing time, since the BP process is directly proportional to the number of hidden units, the number of hidden layers and the number of training points. In the dynamic learning process used for the RBF architecture, the most complex task is the center selection [7]. The key question is how to select centers appropriately from the data set. This implies to calculate the individual contribution to the desired output by each basis vector. In general, the number of all the candidate bases can be very large and an adequate selection procedure involves a global computation with huge memory and time requirements. Though the training of the RBF architecture is fast for a limited number of units, it becomes very low and ineffective for increasing complexity. On the contrary, the memory and time required for training the MLP architecture scale quite well, considering that a few RFs (that might be trained in parallel) are involved.

Fig. 10 shows the result of a computing time analysis, performed on House_16H data, applying a holdout method in which the training set consists of 90% of the available data. Here, the resulting values of the training time versus the number of parameters are jointly plotted for both MLP (black squares) and RBF (black circles) architectures, thus providing a useful comparative insight. Furthermore, for each number of parameters considered, the related number of RFs is also shown in the graph.

The software and hardware platforms are characterized as follows. The used computing framework is *The MathWorks Matlab*™ v. 7; the installed operating system is *Microsoft Windows XP*™ *Professional*; the employed CPU is *Intel Pentium*™ IV, with a clock of 3.20 GHz, a memory cache L1 of 16 KB (data) and 12 K (instructions), a memory cache L2 of 1 MB, and a RAM of 2.00 GB.

As expected, the time increases with the increase of the number of parameters in both training procedures. Note, however, how the MLP architecture requires a shorter training time than the RBF architecture. Further, at least for the reasonable number of

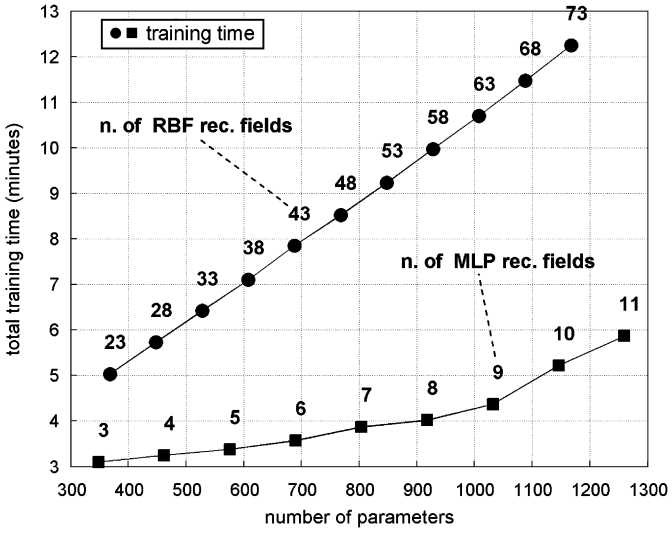

Fig. 10. Total training time versus number of parameters, for RBF systems (●) and MLP systems (■).

parameters taken into account, the MLP architecture is characterized by a higher scalability over networks of growing complexity.

Lastly, the memory requirements in the training phases of MLP and RBF architectures differ for orders of magnitude: about 30–35 MB (MLP system) versus 1.6–1.7 GB (RBF system). This further leads us to appreciate the scalability of the proposed architecture.

In order to provide a formal description of the complexity of each approach, let us consider the asymptotic computational cost, which can be formulated considering the major phases of the development of the networks. In particular, for the MLP system, the blueprint development (Section 4.1) is realized using a standard K-means clustering algorithm applied to the set Y of training outputs, and a CFCM to the set X of training inputs.

As the K-means clustering algorithm is concerned, the asymptotic complexity is $\Theta(N \cdot P \cdot L)$ [45], where N is the number of patterns, P is the number of clusters, and L is the number of iterations taken by the algorithm to converge. Typically, L is fixed in advance, and then the algorithm has a linear time complexity in the size of the data set and in the number of clusters. As regards the clustering phase, in the case of the classic FCM (fuzzy C-means), the asymptotic complexity is $\Theta(N \cdot C^2 \cdot F)$ [17], where C is the number of clusters and F is the input space dimension. Considering the CFCM algorithm, the asymptotic computational cost of a single iteration is the same as in the case of the FCM algorithm. The advantage is that the algorithm is applied to subsets of the data set. For the sake of simplicity, let us assume that the data set is split into P subsets (each of N/P points) by the fuzzy partition, and that each of them is partitioned into C/P clusters [46]. Then C represents also the total number of RFs in the system. Hence, the asymptotic complexity of this phase is $\Theta(N/P \cdot (C/P)^2 \cdot F)$, considering that the P conditional clustering processes are independent, and then they can be performed in parallel.

Second, we consider the parametric learning of the RFs, which comprises a classical BP procedure over a subset of N/P points. The BP process is directly proportional to the number of input neurons (F), the number of hidden units and the number of hidden layers (both fixed), and to the number of training points (N/P). Hence, the learning of each RF costs $\Theta(N/P \cdot F)$ [47]. Finally, we consider the LSE optimization, which mainly involves multiplication and inversion of matrices (due to the term $(\Sigma^T \Sigma)^{-1}$), i.e., a complexity $\Theta(PN^2 + N^3)$ [48].

As regards the dynamic RBF network, the major cost terms are represented by (i) the global search for the location of next radial basis, i.e. $\Theta(N \cdot F \cdot Q)$, and (ii) the LSE optimization $\Theta(QN^2 + N^3)$, where Q is the number of RFs. Then we can conclude that, the asymptotic computational costs of the considered algorithms are:

$$\text{MLP system}: \ [\Theta(N \cdot P) + \Theta(N/P \cdot (C/P)^2 \cdot F) + \Theta(N/P \cdot F)]\Theta(PN^2 + N^3), \tag{8}$$

$$\text{RBF system}: \ [\Theta(N \cdot Q \cdot F)] + \Theta(QN^2 + N^3). \tag{9}$$

Note, first, that P in (8), i.e. number of contexts, is much smaller than Q in (9), i.e. number of RFs. Also, note that the output matrix Σ is sparse in (8) (each column of the matrix has just two non-zero elements), and then both the product and the inversion matrices are actually less expensive in (8) than in (9), considering both temporal and spatial (memory) criteria. Furthermore, it can be observed that in (8) the term C/P is usually of order 1, and then, the major terms are NF/P and NFQ in (8) and (9), respectively. This reveals the better scalability of the proposed systems in terms of N and F.

5.3. Conceptual and spatial criteria in RFs networks

In the proposed model, the use of RFs relies on the decomposition of the input space into sub-regions on the basis of context constraints. This allows global partitioning guided by conceptual criteria, rather than metric ones. However, the RF (MLP) training in each local model is still based on spatial criteria between patterns. To apply this strategy implies obviously that a "global" conceptual criterion and a spatial (i.e., Euclidean) distance between "local" patterns have to be pertinent. Actually, this could not be the case for all the tasks. For example, how a dynamic or noisy function could be globally and locally split with conceptual and spatial criteria, respectively? We can assume that more complex criteria could be more pertinent in such cases. This problem is somewhat present also in dynamic RBF networks. A theoretical investigation needs to be done for this purpose.

To practically illustrate this concept, let us consider the Mackey–Glass (MG) time series. In this case, partitioning of the input space should be performed by using measures different from classical Euclidean spatial distance, as for instance those proposed in [49,50] for RBF networks.

The time series used in the experiments is generated by the chaotic MG differential delay equation (available on http://www.cse.ogi.edu/~ericwan/data.html). In particular, the generation of the series is characterized by a delay parameter τ, usually set to 17 (MG-17 series) or 30 (MG-30 series), which determines the level of chaos, i.e. unpredictability, of the model. It is known, however, that the MG-17 is at the borderline of chaos, as the MG series becomes chaotic if $\tau > 16.8$. To stress the approximation capabilities of the compared systems we used the less common MG-30 series, shown in Fig. 11, with 1500 samples.

As usual in the literature, in order to tackle the problem with spatial criteria, the data are represented in a four-dimensional space where each sample s_n of the series is predicted using four past samples [51]. More specifically, the input and output patterns are constructed using the following feature extraction procedure:

$$\mathbf{x}_n = \{s_{n-v}, s_{n-v-6}, s_{n-v-12}, s_{n-v-18}\}, \quad y_n = s_n, \tag{10}$$

with the parameter $v = 1$ (step-ahead).

In this five-dimensional representation, the training set is composed of the first 90% (i.e., 1350 samples) of the data, and then the test set is formed by the remaining 10% (i.e., 150) of samples.

Fig. 12 shows the results of the experiments. As regards the MLP architecture, performances refer to the average MSE value over 10 trials carried out under the same conditions. Note that in this case the compared systems have quite similar performances in terms of function approximation properties. The results are not particularly good both for the classical RBF network and for our neural architecture. On the other hand, both networks have not been designed for solving these types of problems. Indeed, both networks adopt spatial criteria, in training or positioning RFs, which are not appropriate to time series domains.

Furthermore, the proposed system has been tested in case of noise. Noise affects the task of decomposition in sub-tasks, as well as the local training of each RF. In both architectures, the presence of noise can be tackled only by using specific techniques in clustering and learning processes, depending of the kind of noise.

In order to test the robustness of the system in presence of noisy data, we added Gaussian noise to each pattern of the training set in the time series, keeping the test set noiseless to measure the true prediction error [52]. In particular, the signal to noise ratio (SNR) has been kept low multiplying the generated noise by a constant normalization factor, determined as $0.05 \cdot |\max_n(s_n) - \min_n(s_n)|$, i.e., we considered the 5% of the maximum magnitude of the signal.

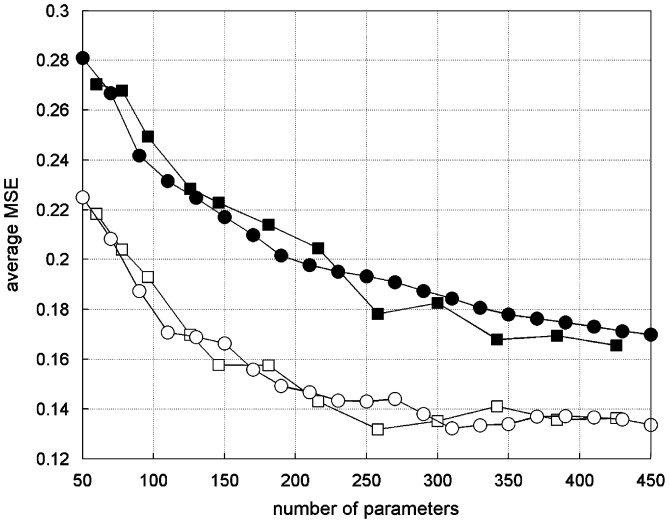

Fig. 12. Mackey–Glass time series, average MSE versus number of parameters, for RBF systems on training (●) and test sets (○), and for MLP systems on training (■) and test sets (□).

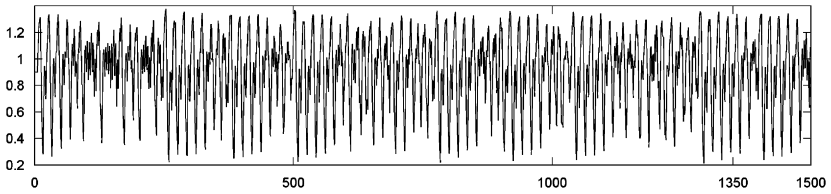

Fig. 11. The Mackey–Glass time series, with $\tau = 30$.

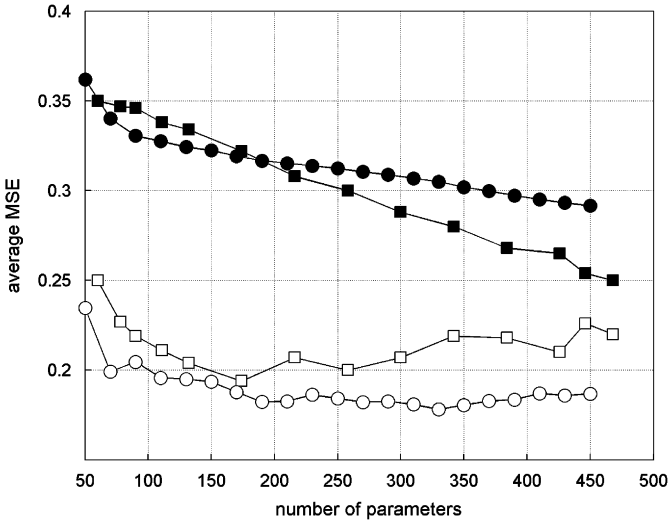

Fig. 13. Mackey–Glass time series with Gaussian noise, average MSE versus number of parameters, for RBF systems on training (●) and test sets (○), and for MLP systems on training (■) and test sets (□).

Table 2
Relative errors of regression algorithms on three benchmark data.

Data set	MLP RF	RPFP	KNN	RULE	MARS	DART
Abalone	0.637	0.675	0.661	0.899	0.683	0.678
Plastic	0.382	0.415	0.475	0.477	0.404	0.432
Fat	0.203	0.667	0.785	0.820	0.305	0.638

Fig. 13 shows the results of the experiments. Here, we can observe that our architecture suffers from overtraining: when the number of parameters increases, the MLP tends to specialize itself too much on the noisy training patterns, loosing its generalization capability. On the other hand, MLPs allow a higher modeling capability than classical RBFs. This is proved by the continuous decrease of the error on the training with the increase of the number of parameters.

5.4. A comparative study with other regression methods

In the previous sections we have discussed the main features and limitations of the proposed paradigm, through a comparison with its counterpart, the RBF neural network. In this section, to show the performance of our regression approach in absolute terms, we compare the performance achieved by our architecture with the selection of regression paradigms discussed in [53]. Here, regression tree induction (DART), instance-based (KNN) learning, rule-based learning (RPFP, RULE) spline-based and partitioning regression (MARS) methodologies are compared with each other by applying a 10-fold cross validation. Most of these methods have been recently developed and outperform earlier algorithms (see [53] for an extensive discussion). Table 2 shows the best average accuracies achieved on *Abalone*, *Plastic* and *Fat* data sets (Publicly available at the Function Approximation Repository of the Bilkent University (http://funapp.cs.bilkent.edu.tr/DataSets/)). The accuracy is measured by using the *relative error* (RE), i.e., the mean absolute distance normalized by the mean absolute distance from the median:

$$RE = \frac{\sum_1^N |\mathrm{out}_k - y_k|}{\sum_1^N |y_k - \mathrm{median}(y)|}, \tag{11}$$

where N is the number of points in the testing set, y_k and out_k are the desired and produced outputs for the k-th input, respectively. This error index was used in [53]. In this subsection, we resorted to this index so as to have the possibility to compare our approach with the regression techniques discussed in [53].

The results show clearly that our approach (viz. MLP RF) outperforms other models in terms of the RE.

6. Conclusions

The concept of receptive fields is the most prominent and ubiquitous conceptual and computational mechanism employed by biological information processing systems. In automated information processing, it is related to an architectural style consisting of a collection of local sub-models that realize some local mappings on some selected regions of the entire modeling domain. Local sub-models are then connected on further levels of processing so as to realize a global mapping. RBF neural networks realize a local character of processing, one of the most used in the literature, as focused on receptive fields. In this study, the use of more complex local receptive fields, precisely RMLPs, has been proposed as a more general and effective way of designing nonlinear mappings. The parametric flexibility of such generalized receptive fields provides a better ability of modeling complex relationships with a lower number of parameters, even when using standard training procedures. A comparative analysis completed for a number of data sets between the standard RBF network and the new proposed architecture has highlighted how the latter outperforms the former both in terms of accuracy and learning time. Further, we have also shown for three commonly exploited data sets that our approach generates models which are more accurate than the ones produced by well-known regression techniques.

Acknowledgment

Support from the Natural Sciences and Engineering Research Council of Canada (NSERC) is gratefully acknowledged.

References

[1] W. Pedrycz, G. Vukovich, System modeling with fuzzy plug-ins, Kybernetes 29 (4) (2000) 473–490.
[2] W. Pedrycz, Conditional fuzzy clustering in the design of radial basis function neural networks, IEEE Trans. Neural Networks 9 (4) (1998) 601–612.
[3] Y. Weiss, S. Edelman, Representation with receptive fields, Technical Report CS93-09, Mathematics & Computer Science, Weizmann Institute of Science, 1993.
[4] E.R. Kandel, J.H. Schwartz, T.M. Jessell, Principles of Neural Science, McGraw-Hill, New York, NY, 2000.
[5] M.W. Levine, J.M. Shefner, Fundamentals of Sensation and Perception, second ed., Brooks/Cole, Pacific Grove, CA, 1991.
[6] M. Han, J. Xi, Efficient clustering of radial basis perceptron neural network for pattern recognition, Pattern Recognition 37 (10) (2004) 2059–2067.
[7] M.J.L. Orr, Introduction to radial basis function networks, Technical Report, Institute for Adaptive and Neural Computation of the Division of Informatics at Edinburgh University, Scotland, UK, 1996.
[8] N.M. Duy, T.T. Cong, Numerical solution of differential equations using multiquadric radial basis function networks, Neural Network 14 (2001) 185–199.
[9] C. Harpham, C.W. Dawson, The effect of different basis functions on a radial basis function network for time series prediction: a comparative study, Neurocomputing 69 (2006) 2161–2170.
[10] S.J. Lee, C.L. Hou, An ART-based construction of RBF networks, IEEE Trans. Neural Network 13 (6) (2002) 1308–1321.
[11] C.C. Lee, P.C. Chung, J.R. Tsai, C.I. Chang, Robust radial basis function neural networks, IEEE Trans. Systems Man Cybernet. B 29 (6) (1999) 674–685.
[12] I. Rojas, H. Pomares, J.L. Bernier, J. Ortega, B. Pino, F.J. Pelayo, A. Prieto, Time series analysis using normalized PG-RBF network with regression weights, Neurocomputing 42 (2002) 267–285.

[13] M. Wallace, N. Tsapatsoulis, S. Kollias, Intelligent initialization of resource allocating RBF networks, Neural Network 18 (2005) 117–122.
[14] E. Blanzieri, Theoretical interpretations and applications of radial basis function networks, Technical Report, Department of Information and Communication Technology, University of Trento, Italy, 2003.
[15] K. Hornik, M. Stinchcombe, H. White, Multilayer feed forward networks are universal approximators, Neural Networks 2 (1989) 359–366.
[16] E. Blanzieri, A. Giordana, Mapping symbolic knowledge into locally receptive field networks, in: M. Gori, G. Soda (Eds.), Topics in Artificial Intelligence, Lecture Notes in Computer Science, vol. 992, Springer, London, UK, 1995, pp. 267–278.
[17] W. Pedrycz, Knowledge-Based Clustering: From Data to Information Granules, John Wiley and Sons, Hoboken, NJ, 2005.
[18] J. Jang, C.T. Sun, Functional equivalence between radial basis function networks and fuzzy inference systems, IEEE Trans. Neural Networks 4 (1993) 156–159.
[19] L. Jiexing, Z. Yun, F. Xi, An improved closest cluster learning algorithm, Control Theory Appl. 17 (2000) 735–738.
[20] Z. Uykan, C. Guzelis, M.E. Celebi, H.N. Koivo, Analysis of input–output clustering for determining centers of RBFN, IEEE Trans. Neural Networks 11 (4) (2000) 851–858.
[21] A. Guillén, J. González, I. Rojas, H. Pomares, L.J. Herrera, O. Valenzuela, A. Prieto, Using fuzzy logic to improve a clustering technique for function approximation, Neurocomputing 70 (16–18) (2007) 2853–2860.
[22] M.M. Brizzotti, A. de Carvalho, The influence of clustering techniques in the RBF networks generalization, in: Proceedings of the IEE Image Processing and its Applications Conference, 1999, pp. 87–92.
[23] A. de Carvalho, M.M. Brizzotti, Combining RBF networks trained by different clustering techniques, Neural Process. Lett. 14 (2001) 227–240.
[24] S.P. Lloyd, Least square quantization in PCM, IEEE Trans. Inf. Theory 28 (2) (1982) 129–137.
[25] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, Wiley-Interscience Publication, New York, 1973.
[26] M.A. Ismail, S.Z. Selim, S.K. Arora, Efficient clustering of multidimensional data, in: Proceedings of the IEEE International Conference on Systems Man and Cybernetics, 1984, pp. 120–123.
[27] M.A. Ismail, M.S. Kamel, Multidimensional data clustering utilizing hybrid search strategies, Pattern Recognition 22 (1989) 75–89.
[28] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkley Symposium in Mathematics, Statistics and Probability, vol. 1, 1967, pp. 281–297.
[29] C. Chinrungrueng, C.H. Sequin, Optimal adaptive K-means algorithm with dynamic adjustment of learning rate, IEEE Trans. Neural Networks 6 (1995) 157–169.
[30] A. Staiano, R. Tagliaferri, W. Pedrycz, Improving RBF networks performance in regression tasks by means of a supervised fuzzy clustering, Neurocomputing 69 (13–15) (2006) 1570–1581.
[31] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of local experts, Neural Comput. 3 (1991) 79–87.
[32] M.I. Jordan, R.A. Jacobs, Hierarchical mixture of experts and the EM algorithm, Neural Comput. 6 (1994) 181–214.
[33] J. Moody, C.J. Darken, Fast learning in network of locally-tuned processing unit, Neural Comput. 1 (1989) 281–294.
[34] D. Kim, K.H. Lee, D. Lee, On cluster validity index for estimation of the optimal number of fuzzy clusters, Pattern Recognition 37 (2004) 2009–2025.
[35] M.H. Hassoun, Fundamentals of Artificial Neural Networks, MIT Press, Cambridge, MA, 1995.
[36] S. Haykin, Neural Networks: A Comprehensive Foundation, Macmillian College Publishing, New York, NY, 1994.
[37] D.L. Chester, Why two hidden layers are better than one, in: Proceedings of the IJCNN '90, vol. 1, Lawrence Erlbaum, Washington, DC, 1990, pp. 265–268.
[38] Z. Obradovic, P. Yan, Small depth polynomial size neural networks, Neural Comput. 2 (1990) 402–404.
[39] S. Tamura, M. Tateishi, Capabilities of a four-layered feedforward neural network: four layers versus three, IEEE Trans. Neural Networks 8 (2) (1997) 251–255.
[40] E. Baum, D. Haussler, What size net gives valid generalization?, Neural Comput. 1 (1989) 151–160.
[41] Y.S. Abu-Mostafa, The Vapnik–Chervonenkis dimension: information versus complexity in learning, Neural Comput. 1 (1989) 312–317.
[42] F. Schwenker, H.A. Kestler, G. Palm, Three learning phases for radial-basis-function networks, Neural Networks 14 (4–5) (2001) 439–458.
[43] S. Chen, C.F.N. Cowan, P.M. Grant, Orthogonal least squares learning algorithm for radial basis function networks, IEEE Trans. Neural Networks 2 (1991) 302–309.
[44] M.T. Hagan, H.B. Demuth, M. Beale, Neural Network Design, PWS Publishing, Boston, MA, 1996.
[45] W.H.E. Day, Complexity theory: an introduction for practitioners of classification, in: P. Arabie, L. Hubert (Eds.), Clustering and Classification, World Scientific Publishing Co., Inc., River Edge, NJ, 1992.
[46] J.F. Kolen, T. Hutcheson, Reducing the time complexity of the fuzzy C-means algorithm, IEEE Trans. Fuzzy Systems 2 (10) (2002).
[47] F. Valafar, O.K. Ersoy, A parallel implementation of backpropagation neural network on maspar mp-1, Electrical and Computer Engineering, Purdue University School of Electrical Engineering, ECE Technical Reports, TR-EE 93-14, 1993.
[48] G. Villard, Exact computation of the determinant and of the inverse of a matrix, Workshop on Complexity, FoCM, Minneapolis, Minnesota, USA, 2002.
[49] S. Elanayar, Y.C. Shin, Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems, IEEE Trans. Neural Networks 5 (1994) 594–603.
[50] R. Zemouri, D. Racoceanu, N. Zerhouni, Recurrent radial basis function network for time-series prediction, Eng. Appl. Artif. Intell. 16 (2003) 453–463.
[51] G.-B. Huang, P. Saratchandran, N. Sundararajan, A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation, IEEE Trans. Neural Networks 1 (16) (2005) 57–67.
[52] K.-R. Müller, A.J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, V.N. Vapnik, Predicting time series with support vector machines, in: W. Gerstner, A. Germond, M. Hasler, J.-D. Nicoud (Eds.), Artificial Neural Networks, Lecture Notes in Computer Science, ICANN'97, vol. 1327, Springer, Berlin, 1997, pp. 999–1004.
[53] İ. Uysal, H.A. Güvenir, Instance-based regression by partitioning feature projections, Appl. Intell. 21 (2004) 57–79.

Mario G.C.A. Cimino received the M.Sc. degree in Computer Engineering and the Ph.D. degree in Information Engineering from the University of Pisa, Italy, in 2003 and 2007, respectively. In 2006, he spent six months as a visiting Ph.D. Student at the Electrical & Computer Engineering Research Facility of the University of Alberta, Edmonton, Canada, under the supervision of Professor W. Pedrycz. His main research interests include relational clustering, robust clustering, neurocomputing, fuzzy rule-based systems, and information systems. He is currently with the Department of Information Engineering of the University of Pisa as a member of the Computational Intelligence Group.

Witold Pedrycz (M'88-SM'94-F'99) received the M.Sc., Ph.D., and D.Sci. degrees from Silesian University of Technology, Gliwice, Poland. He is a Professor and the Canada Research Chair (Computational Intelligence) with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. He is also with the Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland. He is the Editor-in-Chief of Information Sciences. He is the author of 11 research monographs covering various aspects of computational intelligence and software engineering. His current research interests include computational intelligence, fuzzy modeling, knowledge discovery and data mining, fuzzy control including fuzzy controllers, pattern recognition, knowledge-based neural networks, relational computation, and software engineering. He has published numerous papers in these areas. Prof. Pedrycz has been a member of numerous program committees of the IEEE conferences in the area of fuzzy sets and neurocomputing. He is currently an Associate Editor of the IEEE Transactions on Systems Man and Cybernetics, the IEEE Transactions on Fuzzy Systems, and the IEEE Transactions on Neural Networks.

Beatrice Lazzerini is a Full Professor at the Faculty of Engineering of the University of Pisa, Italy. She teaches "Knowledge Engineering and Expert Systems" and "Decision Support Intelligent Systems". She is President of the Specialized Laurea Degree in "Computer Engineering for Enterprise Management". Her main research interests lie in the area of knowledge engineering, with particular emphasis on fuzzy systems, neural networks and evolutionary computation. She has co-authored seven books and has published over 130 papers in international journals and conferences. She is co-editor of the books "Knowledge-Based Intelligent Techniques in Character Recognition," CRC Press, 1999, and "Innovations in ART Neural Networks," Physica-Verlag, 2000. She was involved in several national and international research projects.

Francesco Marcelloni was born in Terni (Umbria) in 1966. He received the Laurea degree in Electronics Engineering and the Ph.D. degree in Computer Engineering from the University of Pisa in 1991 and 1996, respectively. Currently, he is an Associate Professor at the Faculty of Engineering of the University of Pisa. His current research interests include object-oriented software development process, object-oriented models, approximate reasoning, fuzzy rule-based systems, fuzzy clustering algorithms and pattern recognition. He is (co-)author of more than 100 papers in international journals, book chapters and conferences. He has co-founded the Computational Intelligence Group at the Department of Information Engineering of the University of Pisa.