PERGAMON

Neural Networks 11 (1998) 963–973

Neural Networks

Contributed article

An efficient MDL-based construction of RBF networks

Aleš Leonardis[a], Horst Bischof[b,*]

[a]Faculty of Computer and Information Science, University of Ljubljana, SI-1001 Ljubljana, Slovenia
[b]Pattern Recognition and Image Processing Group, Vienna University of Technology, A-1040 Vienna, Austria

Received 2 April 1997; accepted 9 March 1998

Abstract

We propose a method for optimizing the complexity of Radial basis function (RBF) networks. The method involves two procedures: adaptation (training) and selection. The first procedure adaptively changes the locations and the width of the basis functions and trains the linear weights. The selection procedure performs the elimination of the redundant basis functions using an objective function based on the Minimum Description Length (MDL) principle. By iteratively combining these two procedures we achieve a controlled way of training and modifying RBF networks, which balances accuracy, training time, and complexity of the resulting network. We test the proposed method on function approximation and classification tasks, and compare it with some other recently proposed methods. © 1998 Elsevier Science Ltd. All rights reserved.

Keywords: Radial basis functions; Optimizing radial basis function network; Minimum Description Length principle; Function approximation; Heart disease classification

1. Introduction

Radial basis functions (RBFs) have been subject to extensive research over recent years and have successfully been employed to various problem domains (e.g. Moody and Darken, 1989; Lowe, 1989; Girosi et al., 1995; Roy et al., 1995).

In the original proposal of regularization RBF (Poggio and Girosi, 1990) the number of basis functions equals the number of training samples. The basis functions are centered on the training samples and the only unknown parameters are the linear weights which can be determined efficiently by solving the system of linear equations. However, the resulting networks are complex and often ill-conditioned. Generalized RBF networks are designed with fewer nodes than there are samples in the training set, which results in less complex networks. However, using such an approach we somehow have to determine the number of basis functions, their centers, and their widths. Several different strategies have been proposed in the literature, see Haykin (1994) for an overview:

1. a set of samples is randomly selected from the training set and the positions of the centers of the basis functions are set according to these samples (Lowe, 1989). This approach can only produce reasonable results when the training data are distributed in a representative manner;
2. some pre-clustering (grouping) is performed on the training set (e.g. *K*-means clustering), and the centers of the clusters are used as the centers for the basis functions (Moody and Darken, 1989). Since this clustering is performed without the knowledge of the weights of the output nodes, it is very likely that the selection of the centers is suboptimal with respect to the accuracy of the final result;
3. a gradient descent type learning procedure is used to determine the weights of the output nodes, centers and the width of the basis functions (Lowe, 1989). Convergence to a global minimum cannot be guaranteed since the problem is non-linear with respect to the centers and widths of the basis functions.

All these approaches have various shortcomings. The common and the most crucial one is that the number of basis functions has to be given a priori. This is an instance of the model (order) selection problem. Simply stated, we would like to find a model as simple as possible that fits a given data set with sufficient accuracy, and more importantly, generalizes to unseen data. Many different approaches have been used, which can be categorized based on the prevailing aspect of the method as follows.

* Corresponding author. Tel.: (+43) 1 58801 4478; fax: (+43) 1 5054668; e-mail: bis@prip.tuwien.ac.at

Cross validation: many different networks (varying in complexity) are trained and then tested on an independent validation set, or if such a set is not available (i.e. all data is used for training), one uses methods such as leave-one-out cross validation (Stone, 1974). These procedures are computationally very demanding and/or require additional data withheld from the training samples.

Complexity criteria: in order to avoid the use of a validation set and to assess the generalization performance, various criteria of the form

$$\text{prediction error} = \text{training error} + \eta \text{ complexity term}$$

have been proposed. These include Akaike Information Criterion (Akaike, 1973), Generalized Prediction Error (Moody, 1992), Vapnik's Guaranteed Risk Minimization (Vapnik, 1982), Minimum Description Length (MDL) Principle (Rissanen, 1989), Maximum Penalized Likelihood (Sardo and Kittler, 1996). Also, Bayesian methods (MacKay, 1992a,b) for model selection belong to this category, where the complexity term is based on the ratio between posterior and prior of the weight distribution. All these methods require training of many networks and, therefore, are computationally demanding.

Regularization: these methods modify the error function used for training the network, i.e.

$$\text{error} = \text{training error} + \lambda \text{ penalty}$$

in order to force smooth mappings (Orr, 1995). These methods include weight decay (Hinton, 1989; Weigend et al., 1990), soft weight sharing (Nowlan and Hinton, 1992), curvature driven smoothing (Bishop, 1993), flat minima search (Hochreiter and Schmidhuber, 1997) (which is based on the MDL Principle). Closely related are methods of early stopping and training with noise (Bishop, 1995). To reduce the size of the network, pruning methods are usually used. In order to find the regularization parameters either cross-validation or Bayesian techniques are used. Very often the form of the penalty term can be derived from prior weight distributions by Bayesian methods.

Pruning/growing methods: these methods adapt the structure of the network during training. We can distinguish between the constructive methods (e.g. Fritzke, 1994; Fahlman and Lebiere, 1990; Platt, 1991), which incrementally build the network, and pruning methods (e.g. Le Cun et al., 1988; Mozer and Smolensky, 1989; Hassibi and Stork, 1993) which start with a complex network and remove units and/or weights. Sometimes pruning and growing methods are also iteratively combined together (Yingwei et al., 1997). These methods are usually used in conjunction with other methods mentioned earlier (e.g. Orr, 1995). Pruning and growing methods are more efficient than cross-validation techniques; however, they very often make restrictive assumptions so that the resulting networks are suboptimal. Also, evolutionary approaches to the design of neural networks are in this category (e.g. Whitehead and Choate, 1995). The basic idea is to use a genetic algorithm to perform a search on the space of architectures. The main drawback of the evolutionary algorithms is their computational complexity.

Our method can be characterized as a pruning method, starting with an ''overly complex'' network and then gradually reducing it, arriving at a simpler one. The way that we achieve the reduction in the complexity of a network can be related to the principle of simplicity, which has a long history in psychology (Gestalt principles), and whose formalization led in information theory to the method of Minimum Description Length (MDL) (Rissanen, 1984; Zemel and Hinton, 1995). According to the MDL principle, those RBFs are selected from a set of initial RBFs that describe the training data with the shortest possible encoding. However, instead of training many networks and then selecting the best one (according to the MDL principle), we perform the selection during the training, achieving a computationally efficient procedure.

The structure of this paper is as follows. In Section 2 we introduce the notations for RBFs and define some quantities which are required in our algorithm. In Section 3 we derive the selection mechanism from the MDL. Section 4 explains how to combine iteratively the training procedure and the selection procedure to achieve an overall efficient method. Section 5 presents the results of our algorithm applied to the problem of approximating functions. We also tested the method on a well-known example from the medical domain and compared our results with those recently published. Section 6 presents a conclusion and outlines avenues for further research.

2. Network

In order to simplify the notation we use axes-aligned Gaussian RBF networks with a single linear output unit.[1] Let us consider an RBF-network as a function approximator:

$$\begin{aligned} y(\boldsymbol{x}) &= w_0 + \sum_{i=1}^{M} w_i r_i(\boldsymbol{x}) = w_0 + \sum_{i=1}^{M} w_i \, \mathrm{e}^{-(\|\boldsymbol{x}-\boldsymbol{c}_i\|^2/\boldsymbol{s}_i^2)} \\ &= w_0 + \sum_{i=1}^{M} w_i \, \mathrm{e}^{-\sum_{j=1}^{d} [(x_j - c_{ji})^2/s_{ji}^2]} \end{aligned} \tag{1}$$

where c_i and s_i are the center and widths of the ith basis function respectively. We train the network to approximate an unknown function f given a (possibly noisy) training set $TS = \{(\boldsymbol{x}^{(p)}, f(\boldsymbol{x}^{(p)})) | 1 \le p \le q, \boldsymbol{x}^{(p)}, f(\boldsymbol{x}^{(p)}) \in \mathbb{R}^d\}$.

[1] The generalization to other basis functions and multiple output units is straightforward.

2.1. Adapting the network

Given the number of basis functions, we can train the whole network by minimizing an error function, e.g. the usual quadratic error function (Eq. (2))

$$E = \sum_{p=1}^{q} [f(\boldsymbol{x}^{(p)}) - y(\boldsymbol{x}^{(p)})]^2 \tag{2}$$

Training can be performed with a suitable learning algorithm, e.g. gradient descent, Levenberg–Marquardt, Extended Kalman filter, etc. In general, since we also train the centers and widths of the basis functions, we cannot expect that a training algorithm will find a global optimal solution. In our experiments we used gradient descent and Levenberg–Marquardt optimization methods.

Let us define a few quantities which can be derived from the network and which we need to evaluate the complexity of the network in the selection procedure. Given a sample $\boldsymbol{x}^{(p)}$, the influence of the basis function i on the network's output is just the difference between the output in the presence of the basis function and the absence of the basis function:

$$I(r_i(\boldsymbol{x}^{(p)})) = |w_i r_i(\boldsymbol{x}^{(p)})| \tag{3}$$

Using this measure we can define several other quantities that we need for the selection procedure. The domain of the basis function is given by

$$R_i = \{p | I(r_i(\boldsymbol{x}^{(p)})) > \Theta\} \tag{4}$$

where Θ is the threshold which defines when we count a sample to be encoded by the basis function.[2] The number of samples encoded jointly by the basis functions i and j is given by

$$n_{ij} = |R_i \cap R_j| \tag{5}$$

The error associated with a basis function is defined as

$$\xi_i = \sum_{p \in R_i} [y(\boldsymbol{x}^{(p)}) - f(\boldsymbol{x}^{(p)})]^2 = \sum_{p \in R_i} \xi_i^{(p)} \tag{6}$$

and the error in the overlapping area of two basis functions is given by

$$\xi_{ij} = \sum_{p \in R_i \cap R_j} [y(\boldsymbol{x}^{(p)}) - f(\boldsymbol{x}^{(p)})]^2 \tag{7}$$

One should note that these quantities can easily be calculated while performing an adaptation step, resulting in almost no additional computational costs.

3. Selection

In this section, we elaborate on the RBF-selection procedure which has the task of removing redundant basis functions by the MDL principle. This involves the design of an objective function which encompasses the information about the competing RBF units and the design of an optimization procedure which selects a set of units accordingly.

The task of selection can simply be formulated as:

> Given a set of RBF nodes together with their corresponding parameters (centers, widths, weights to output units) combined in $\boldsymbol{a}_i$, and domains $\{(\boldsymbol{a}_i\}, R_i)_{i=1,M}\}$, select an optimal subset of RBFs $\{(\boldsymbol{a}_i^*, R_i^*)_{i=1,n_R}\}$ which comprise the final result such that $n_R \leq M$.

The core of the problem is how to define optimality criteria for choosing the ''best'' network configuration. Intuitively, this reduction in the complexity of a representation coincides with a general notion of simplicity which has a long history (Leclerc, 1989). The simplicity principle has been formalized in the framework of information theory. Shannon (1948) revealed the relation between the probability theory and the shortest encoding. Further formalization of this principle in information theory led to the principle of MDL (Rissanen, 1983). For a practical application, however, it is sufficient to know that the minimum-length interpretation is valid when we have a coding language, which does not necessarily have to be binary. In our case the coding language is given by the RBF-network.

Prior to modeling a function with a network, the function can only be given by specifying the samples. After building a network some of the samples, or possibly all of them, can be described in terms of the network. However, the complexity of a network may vary, i.e. the network may contain more or less nodes. Let vector $\boldsymbol{m}^{\mathrm{T}} = [m_1, m_2, \ldots, m_M]$ denote a set of RBF nodes, where m_i is a presence-variable having the value 1 for the presence of a node and 0 for its absence in the final network, and M is the number of all initial RBF nodes. The length of encoding of a function L_{function} can be given as the sum of two terms:

$$L_{\text{function}}(\boldsymbol{m}) = L_{\text{samples}}(\boldsymbol{m}) + L_{\text{network}}(\boldsymbol{m}) \tag{8}$$

$L_{\text{samples}}(\boldsymbol{m})$ is the length of encoding of individual samples that are not described by the network, and $L_{\text{network}}(\boldsymbol{m})$ is the length of encoding of samples (data) described by the RBF network. The idea is to select a subset of RBF nodes that would yield the shortest description length. In other words, we should tend to maximize the efficiency of the description, defined as

$$E = 1 - \frac{L_{\text{function}}(\boldsymbol{m})}{L_{\text{samples}}(\boldsymbol{0})} \tag{9}$$

where $L_{\text{samples}}(\mathbf{0})$ denotes the length of encoding of the input data in the absence of the network.

Alternatively, we can define a quantity S which represents the savings in the length of encoding of samples in the presence of a network[3]

[2] This threshold can be defined either using knowledge about the expected error, or via the desired quantization of the final result.

[3] A similar definition was also proposed by Fua and Hanson (1989).

$S =$ length of encoding of the data in the absence of a network $-$ length of encoding of the data with a network

$$= L_{\text{samples}}(\mathbf{0}) - L_{\text{function}}(\boldsymbol{m}) \tag{10}$$

The question is how to translate the above equations into our particular case of an RBF neural network. Remember that we can identify the following three terms for the ith RBF node in the network:

1. a set of input samples R_i, which activate the RBF node. They represent the domain of the node and encompass $n_i = |R_i|$ samples;
2. the set of parameters of the node $\boldsymbol{a}_i$ (location, width, weights to output units). N_i denotes the cardinality of this set;
3. the goodness-of-fit measure ξ_i, which gives the difference between the true values of the function and the approximations produced by the network over the domain R_i and projected onto the ith node.

Analogous to Eq. (8) we can write

$$L_{\text{function}}(\boldsymbol{m}) = K_1[n_{\text{all}} - n(\boldsymbol{m})] + K_2\xi(\boldsymbol{m}) + K_3N(\boldsymbol{m}) \tag{11}$$

where n_{all} denotes the number of all sample data points in the training set and $n(\boldsymbol{m})$ the number of data points that are covered by the network i.e. $n(\boldsymbol{m}) = |\cup_{i=1}^{M} R_i|$ (see also discussion in Section 3.2 which describes how outliers can be handled). $N(\boldsymbol{m})$ specifies the number of parameters which are needed to describe the network and $\xi(\boldsymbol{m})$ gives the deviation between the data and the approximation produced by the network. K_1, K_2, K_3 are weights which can be determined on a purely information-theoretical basis (in terms of bits), or they can be adjusted in order to express the preference for a particular type of description (see Section 3.1).

Now we can state the task as follows: find $\hat{\boldsymbol{m}}$ such that

$$L_{\text{function}}(\hat{\boldsymbol{m}}) = \min_{\boldsymbol{m}} L_{\text{function}}(\boldsymbol{m}) \tag{12}$$

Since n_{all} is constant, minimization of Eq. (11) is equivalent to maximizing the expression

$$F(\hat{\boldsymbol{m}}) = \max_{\boldsymbol{m}} F(\boldsymbol{m}) = K_1 n(\boldsymbol{m}) - K_2\xi(\boldsymbol{m}) - K_3N(\boldsymbol{m}) \tag{13}$$

This equation supports our intuitive thinking that an encoding is efficient if

- the number of data samples that an RBF encompasses is large,
- the deviations between the data and the approximation are low,
- while at the same time the number of network parameters is small.

Expanding Eq. (13) in terms of individual basis functions gives us

$$\begin{aligned} F(\hat{\boldsymbol{m}}) = \max_{\boldsymbol{m}} F(\boldsymbol{m}) \\ &= K_1(n_1m_1 + \ldots + n_Mm_M) \\ &- K_2(\xi_1m_1 + \ldots + \xi_Mm_M) - K_3(N_1m_1 + \ldots + N_Mm_M) \\ &- K_1(n_{12}m_1m_2 + \ldots + n_{1M}m_1m_M + \ldots + n_{(M-1)M}m_{(M-1)}m_M) \\ &+ K_2(\xi_{12}m_1m_2 + \xi_{13}m_1m_3 + \ldots + \xi_{1M}m_1m_M + \ldots \\ &+ \xi_{(M-1)M}m_{(M-1)}m_M) \\ &+ K_1(n_{123}m_1m_2m_3 + \ldots + n_{(M-2)(M-1)M}m_{(M-2)}m_{(M-1)}m_M) \\ &- K_2(\xi_{123}m_1m_2m_3 + \xi_{(M-2)(M-1)M}m_{(M-2)}m_{(M-1)}m_M)\ldots \end{aligned} \tag{14}$$

If we consider only pairwise overlaps (see also Pentland (1990)), we can write Eq. (14) more compactly in the following form:[4]

$$F(\boldsymbol{m}) = [\, m_1 \ \ldots \ m_i \ \ldots \ m_M \,] \begin{bmatrix} c_{11} & \ldots & c_{1i} & \ldots & c_{1M} \\ \vdots & & \vdots & & \vdots \\ c_{i1} & \ldots & c_{ii} & \ldots & c_{iM} \\ \vdots & & \vdots & & \vdots \\ c_{M1} & \ldots & c_{Mi} & \ldots & c_{MM} \end{bmatrix} \begin{bmatrix} m_1 \\ \vdots \\ m_i \\ \vdots \\ m_M \end{bmatrix} \tag{15}$$

or in short

$$F(\boldsymbol{m}) = \boldsymbol{m}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{m} \tag{16}$$

The diagonal terms of the matrix $\boldsymbol{C}$ express the cost–benefit value for a particular basis function i

$$c_{ii} = K_1n_i - K_2\xi_i - K_3N_i \tag{17}$$

while the off-diagonal terms handle the interaction between the overlapping basis functions

$$c_{ij} = \frac{-K_1n_{ij} + K_2\xi_{ij}}{2} \tag{18}$$

From the computational point of view, it is important to notice that the matrix $\boldsymbol{C}$ is symmetric, and depending on the overlap of the basis functions, it can be sparse or banded. All these properties of the matrix $\boldsymbol{C}$ can be used to reduce the computations needed to calculate the value of $F(\boldsymbol{m})$.

3.1. Parameters in the objective function

There are three parameters (K_1, K_2, K_3) involved in the computation of the objective function (Eqs. (14) and (15)).

[4] Note that we have made use of the fact that $m_i \in \{0, 1\}$ and therefore $m_i = m_i^2$.

These parameters are weights which can be determined on a purely information-theoretical basis (in terms of bits), or they can be adjusted in order to express the preference for a particular type of description. In general, K_1 is the average number of bits which are needed to encode a sample when it is not encoded by the network, K_2 is related (as shown below) to the average number of bits needed to encode a residual value in the presence of a network, and K_3 is the average cost of encoding a parameter of the network.

Since we are only interested in the relative comparison of possible descriptions, we can set K_1 which weights the number of data points to 1 and normalize K_2 and K_3 relative to it. If we assume for a moment that the error equals 0, K_3 determines the minimum number of samples that a basis function has to encompass in order to be preferred over a point-wise description. In our case of axis-aligned basis functions we can set $K_3 = 1$ if we assume that the data and the weights are given with the same accuracy (i.e. they require the same number of bits).

In order to determine the value of K_2, let us suppose that deviations of the samples from the approximations produced by the network are modeled as zero-mean Gaussian noise with variance σ^2. The length of encoding of a discrete random (IID) process, i.e. a sequence of random variables $x_1, x_2, \ldots, x_n$ with the probability distribution $p(x) = N(0, \sigma)$, finely quantized using intervals of fixed width t, is determined as

$$L_{\text{deviations}} = -\log_2\left[\prod_{i=1}^{n} p(x_i)t\right] = -\sum_{i=1}^{n}\log_2[p(x_i)t] = -\sum_{i=1}^{n}\log_2\left(\frac{t}{\sqrt{2\pi}\sigma}\exp^{-(1/2)(x_i^2/\sigma^2)}\right) \tag{19}$$

More specifically:

$$L_{\text{deviations}} = -\log_2 \exp^{-(1/2)\sum_{i=1}^{n}(x_i^2/\sigma^2)} + n(\log_2\sigma + \tfrac{1}{2}\log_2 2\pi - \log_2 t) \tag{20}$$

We estimate $\sigma^2 = (1/n)\sum_{i=1}^{n} x_i^2$. The first term on the right side of Eq. (20) can be expressed as

$$-\log_2 \exp^{-(\sum_{i=1}^{n} x_i^2)/2\sigma^2} = \frac{\sum_{i=1}^{n} x_i^2}{2\sigma^2}\log_2 \mathrm{e} = \frac{n}{2}\log_2 \mathrm{e} \tag{21}$$

Eq. (19) can now be written as

$$L_{\text{deviations}} = \frac{n}{2}\log_2 \mathrm{e} + n(\log_2\sigma + \tfrac{1}{2}\log_2 2\pi - \log_2 t) = n\left[\frac{\log_2(\sigma^2 2\pi\, \mathrm{e}/t^2)}{2}\right] \tag{22}$$

The second term in Eq. (17), $K_2\xi_i = K_2 n\sigma^2$, handles the deviation between the samples and the approximations produced by the network. By comparing the term $K_2\xi_i$ to $L_{\text{deviations}}$ (Eq. (22)), we obtain the value for K_2:

$$K_2 = \frac{\log_2(\sigma^2 2\pi\, \mathrm{e}/t^2)}{2\sigma^2} \tag{23}$$

Note that the value of K_2 depends on the standard deviation of the noise distribution and decreases with an increase of the noise level σ. However, when σ is known to be within a range of values, we can calculate an approximate value for K_2 and keep it constant for that particular case. Since we normalize the parameters with respect to K_1, we need to divide the above-obtained value K_2 by the number of bits needed to specify a training sample. It is also important to note that the algorithm is stable with respect to K_2. The value of K_2 calculated by Eq. (23) should be considered as an upper bound, and the experiments have shown that the results stay stable if we use values for K_2 which are smaller than this bound.

3.2. Remarks

Assuming the same cost for each basis function (i.e. the same prior probability), the cost term N_i is a constant. However, we can incorporate a measure on the required precision of the parameters as proposed by Hochreiter and Schmidhuber (1997) in their flat minima search algorithm, which gives the basis functions that can be specified with less precision a higher probability of selection. In the case when we have a network with different types of basis function (i.e. spherical Gaussian, axes-aligned Gaussians, etc.), the term N_i can be used to reflect the different complexities. The selection procedure, in this case, selects more complex basis functions only when they result in a shorter description.

The approximation that we make by considering only pairwise overlaps of basis functions in Eq. (15) can be justified by the way that we combine the selection with the adaptation procedure (see Section 4).

As can be seen from the derivation of the selection procedure, we also have the possibility to encode a sample individually and not through the network (e.g. in the case of an extreme outlier point). To realize this option we need to restrict the domain of the basis function to data points which result in bounded errors and to use a training procedure based on robust statistics (e.g. Chen and Jain, 1994; Liano, 1996).

3.3. Solving the optimization problem

We have formulated the problem in such a way that its solution corresponds to the global extremum of the objective function. Maximization of the objective function $F(\boldsymbol{m})$ belongs to the class of combinatorial optimization problems (quadratic Boolean problem). Since the number of possible solutions increases exponentially with the size of the problem, it is usually not tractable to explore them exhaustively. Thus the exact solution has to be sacrificed to obtain a practical one. Various methods have been proposed for finding a ''global extreme'' of a class of non-linear objective functions. Among these methods are winner-takes-all strategy, simulated annealing, microcanonical annealing, mean field annealing, Hopfield networks, continuation methods, and genetic algorithms (Cichocki and Unbehauen, 1993). In the experiments reported below we have used both a winner-takes-all method and a Tabu search algorithm (Glover and Laguna, 1993) without any significant difference in the number and parameters of selected basis functions.

4. Adaptation and selection combined in an iterative scheme

In the previous sections we have explained how we adapt an RBF network, and how we perform the selection procedure. In this section we combine these two procedures in order to form a computationally efficient algorithm for RBF-network design. We now describe the complete algorithm.

(1) *Initialization*: for each sample $\boldsymbol{x}^{(p)} \in TS$ in the training set generate a basis function r_p with the center $\boldsymbol{c}_p = \boldsymbol{x}^{(p)}$ and width $s_{ip} = |c_{ip} - c_{ij}|/1.25$ where $j = \arg\min_{k \neq p} |c_{ip} - c_{ik}|$.[5] The output weights are set by a regularization algorithm in order to avoid the ill-conditioning of the matrix inversion.

(2) *Adaptation*: train $\boldsymbol{c}_k$, $\boldsymbol{s}_k$, and $\boldsymbol{w}_k$ for a fixed number of steps with a training algorithm, e.g. gradient descent, Levenberg–Marquardt.[6]

(3) *Selection*:

(a) calculate $\boldsymbol{C}$ using Eqs. (17) and (18);
(b) optimize $F(\boldsymbol{m})$ with respect to $\boldsymbol{m}$ (Eq. (16));
(c) if $m_i = 0$ remove the basis function i from the network.

(4) If the selection procedure has not removed any of the basis functions during the last selection, and the changes of weights are small, terminate the algorithm, otherwise go to step (2).

This iterative approach is a very controlled way of removing redundant basis functions. Since the selection is performed based on the relative competition among RBFs, only those basis functions are removed that cause a high error and overlap with other basis functions which better approximate the data. The basis functions which remain in the network are then further adapted by the training procedure. To achieve a proper selection it is not necessary to train the network to convergence at each step. When the selection procedure is invoked, only those basis functions are removed where others can compensate for their omission. If the selection procedure is called too ''early'', no basis functions will be removed, causing only additional computational overhead. On the other hand, if we call the selection procedure too ''late'' we are unnecessarily training some basis functions which will then be removed, again causing only additional computational overhead. In this sense, the invocation of the selection procedure is independent of the stage of the training and, therefore, it is not critical when it is called.

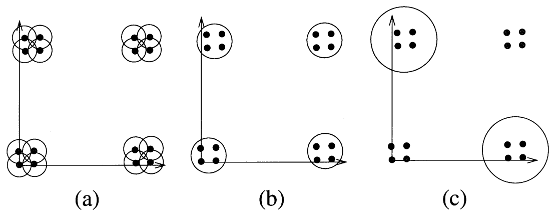


Fig. 1. Demonstration of the algorithm on a modified XOR problem.

Let us illustrate the behavior of the algorithm with a simple (hand drawn) example. Fig. 1 shows a modified XOR problem. In the initialization step a basis function is generated for each sample, Fig. 1(a). Then the basis functions are adapted (in position and widths) by the training algorithm, and the selection procedure selects one basis function for each cluster. After further training we obtain Fig. 1(b). In one further iteration of the algorithm two more basis functions are eliminated (corresponding to the samples with output 0, because their influence on the network output is negligible), Fig. 1(c). After that, no more basis functions are eliminated. Thus, we end up with a network of two basis functions which solves this problem.

5. Experimental results

In order to test the proposed method we applied it to the problem of approximating functions (Orr, 1995), and to a classification problem from the medical domain which was recently investigated by RBF-like networks (Roy et al., 1995).

5.1. Function approximation

The function approximation results were obtained with the following parameters: the Levenberg–Marquardt algorithm was used for training the network and a winner-takes-all algorithm for the selection procedure; $\Theta = 0.1$, $K_1 = 1$, and $K_3 = 1$. K_2 was calculated using Eq. (23) assuming $\sigma = 0.1$, $t = 0.01$, and 32-bit accuracy on the training samples, which resulted in $K_2 = 16$. The same values were used in all function approximation experiments (though sometimes the noise level is different, demonstrating the robustness of the method with respect to variations of this parameter).

The first experiment involved constructing a network for function approximation; we have used the Hermite polynomial from Orr (1995). The training set consists of 84

[5] This initialization ensures that the basis functions are partially overlapping. The constant 1.25 was set such that the activation of neighboring basis functions is at least 0.2.

[6] It is important to note that we do not train the network to convergence, usually only a few epochs are sufficient, e.g. approx 10 for a steepest descent method (see following discussion).

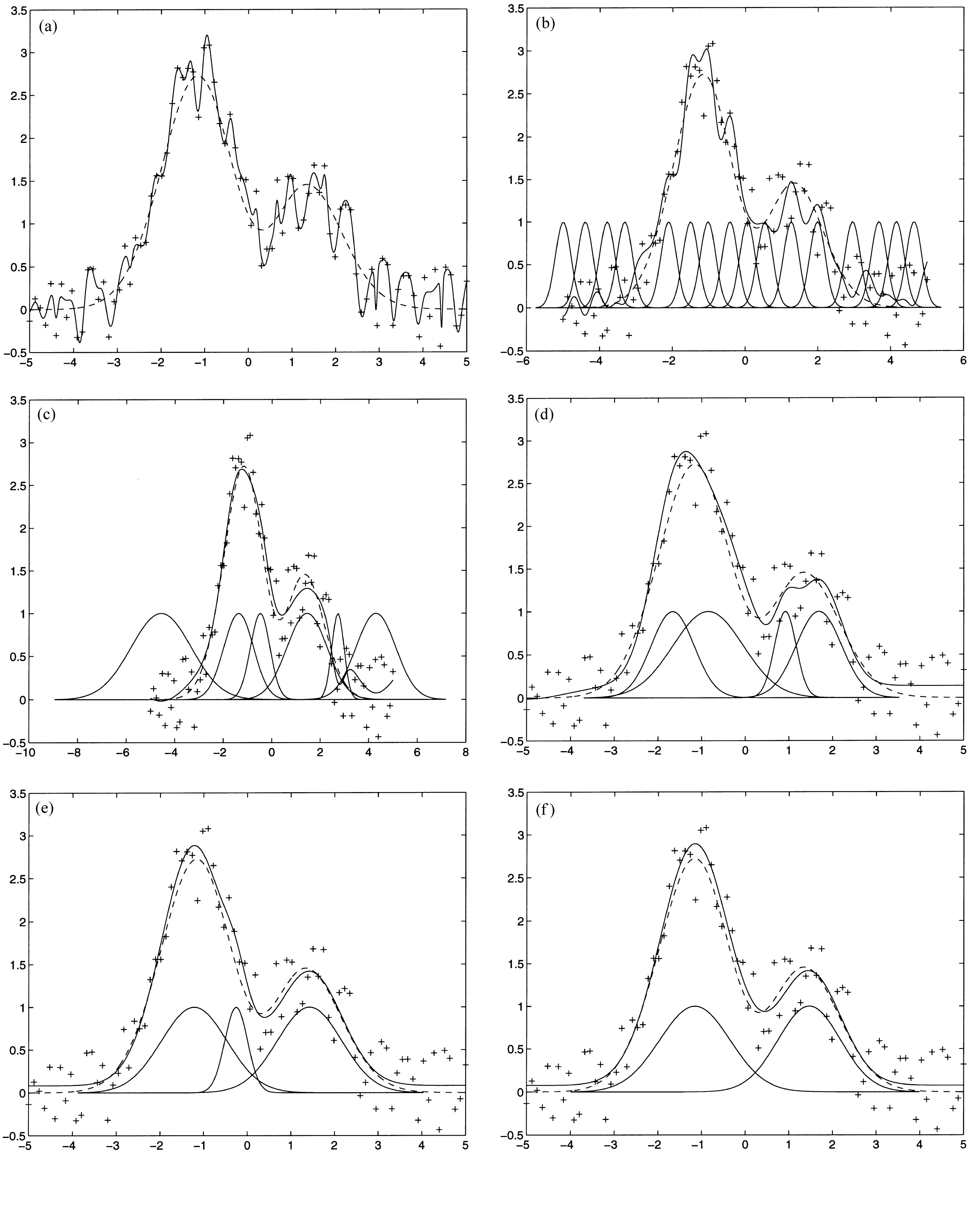


Fig. 2. Results on Hermite polynomial with superimposed basis functions: " + " denotes training samples, dashed curve denotes original function, and solid curve denotes the approximated function. (a) Initialized network; (b) 1st selection; (c) 2nd selection; (d) 3rd selection; (e) 4th selection; (f) final result.

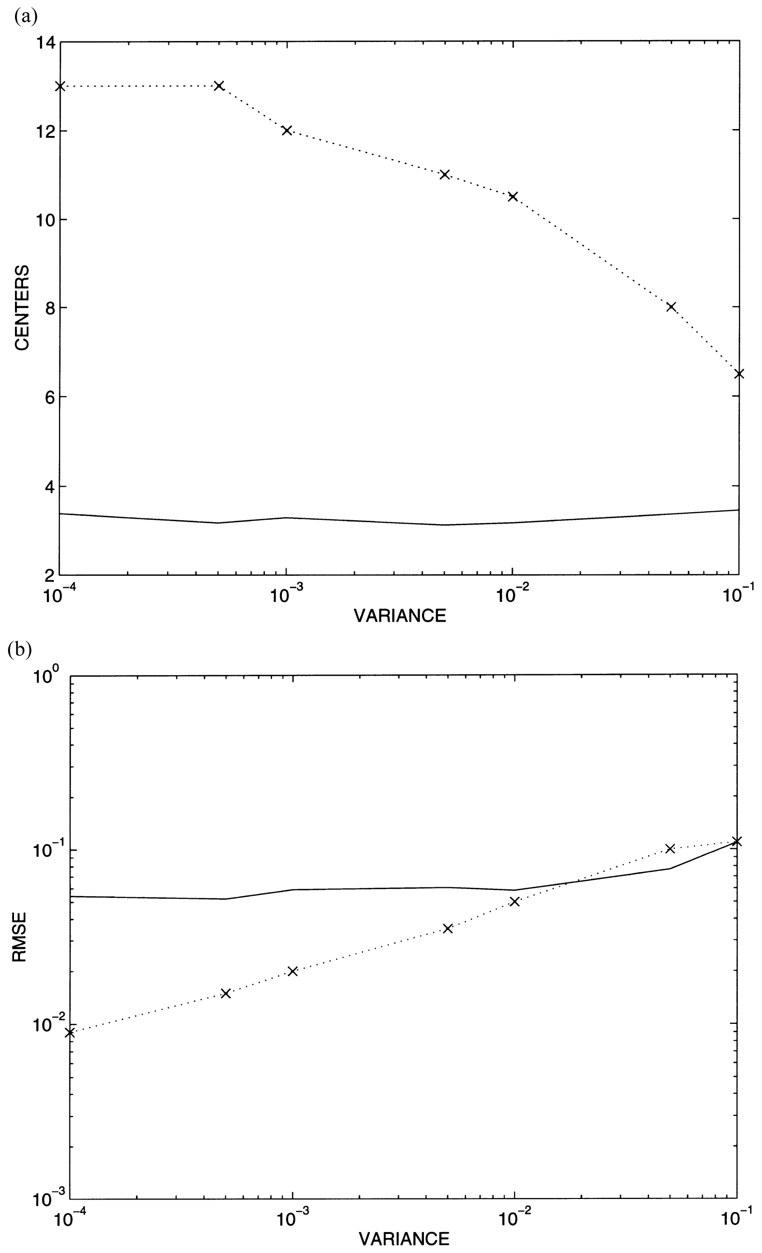


Fig. 3. (a) The number of selected centers and (b) the fit error as a function of the noise level (averaged over 100 runs) of our algorithm (solid curve) and Forward selection of Orr (dotted curve).

samples taken from the function with added noise with a uniform distribution in the range of $[-0.5, +0.5]$. The selection converged at two basis functions after four selection steps. Fig. 2 shows the intermediate results during the run of the algorithm. This figure shows the training samples, the true function, the approximated function, and the basis functions currently in the network.

In the second set of experiments we compared our algorithm with the one by Orr (1995). We have used the same experimental setting: the number of training samples is 40, and the test set is 200 uniformly spaced noiseless samples in the range $-4 \leq x \leq 4$. The results are averaged over 100 runs for each of the several different noise levels in the training data. The randomly chosen 40 training samples in each run were used to initialize the algorithm. Fig. 3 shows how the average of the number of selected centers over 100 runs and the root-mean-squared error (of the fit over the test set) vary with the noise variance.[7]

As can be seen from the figures, our algorithm is very robust with respect to the noise level. We consistently end up on the average with approximately 3.3 basis functions,

[7] The results obtained by the algorithm proposed by Orr was read off from Fig. 4 in Orr (1995).

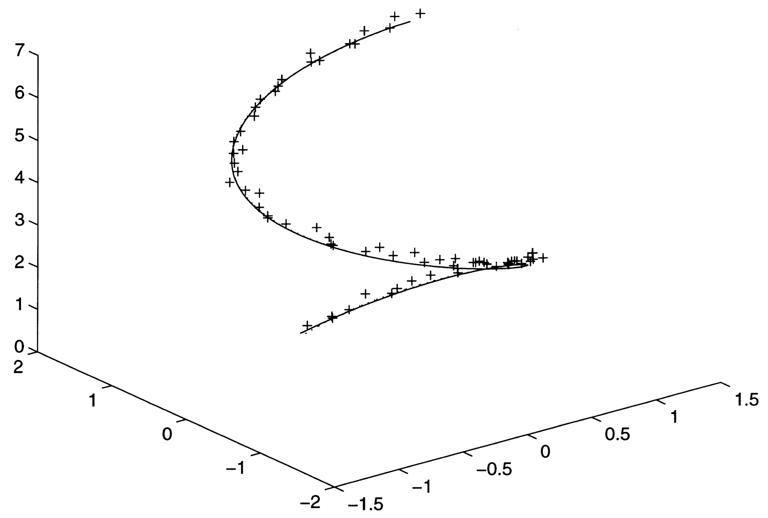


Fig. 4. Result of approximating noisy $f(z) = [\sin(z), \cos(z)]^T$ (dotted curve): " + " denotes training samples and solid curve denotes the approximated function; the dotted curve is the original function.

whereas the best results reported by Orr are six basis functions. The higher error for the low noise case can be attributed to the training algorithm and the fact that we use less basis functions.

Yingwei et al. (1997) also report on the Hermite polynomial approximation problem. Using a noiseless training set, their algorithm resulted in seven basis functions, with a similar approximation error as was reported by Orr (1995).

Fig. 4 shows a result on approximating a function with two outputs $f(z) = [\sin(z), \cos(z)]^T$ with zero mean Gaussian noise with $\sigma = 0.07$. In this case we ended up with four basis functions after three selection steps. One can hardly see any difference between the original and the approximated function (only at the beginning of the function is there a small deviation).

These results demonstrate that our algorithm is able to find a good compromise between the number of basis functions and the approximation error.

5.2. Classification

The next example deals with a heart disease diagnosis task. Roy et al. (1995) have performed an extensive comparison of various algorithms on this database. The best results are achieved with a ‘‘Gaussian Masking (GM)’’ algorithm with a recognition rate of 81.82% with 24 basis functions. Other RBF networks that were also tested performed considerably worse on this data set. The data set contains 292 13-dimensional samples, classified in two classes. We have normalized the features so that they lie in the range between [0…1]. From this set we randomly selected 100 times 50% of the samples for training, the rest was used to test the generalization of the network. This time we have used gradient descent as the training algorithm and Tabu search for the selection procedure. The parameters of our algorithm were set as follows: $\Theta = 0.1$, $K_1 = 1$, and $K_3 = 1$; K_2 was set to 1, because the cost of an error in a two-class classification problem is 1.

Fig. 5 shows the number of selected basis functions as well as the recognition rate of the algorithm over 100 runs. On the average we ended up with 15.49 centers with a standard deviation of 2.14, and a mean recognition rate of 85.98% with a standard deviation of 2.69%.

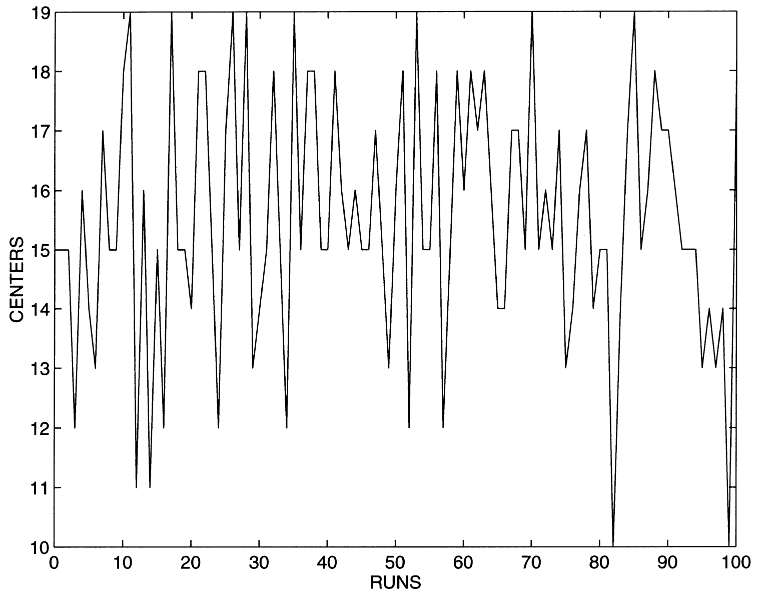
This result is considerably better in terms of the number of basis functions, as well as in terms of the recognition rate, than all previously published results.

6. Conclusion

We presented a method which, starting from an initially high number of basis functions in an RBF network, through an iterative procedure of training, and selection, achieves a compact network. Our approach effectively and systematically overcomes the problems of how many RBFs to use and where to place them. We have demonstrated that our method performs better than some recently published methods for RBF construction.

The algorithm, which we have presented can also be used for types of model other than neural networks. Originally, a very similar algorithm was proposed for parametric models (i.e. surfaces, lines, circles, etc.) (Leonardis et al., 1995; see also Stricker and Leonardis (1995)). We have also used a modified version of the algorithm to compute coefficients of principal components from noisy images (Leonardis and Bischof, 1996). This demonstrates that the approach is quite general and can easily be adapted to other types of network. For example, we have used almost the same algorithm for multi-layer perceptrons (Bischof and Leonardis, 1998). In addition, we are currently working on adapting the algorithm for several other networks. This includes Gaussian

(a)



(b)

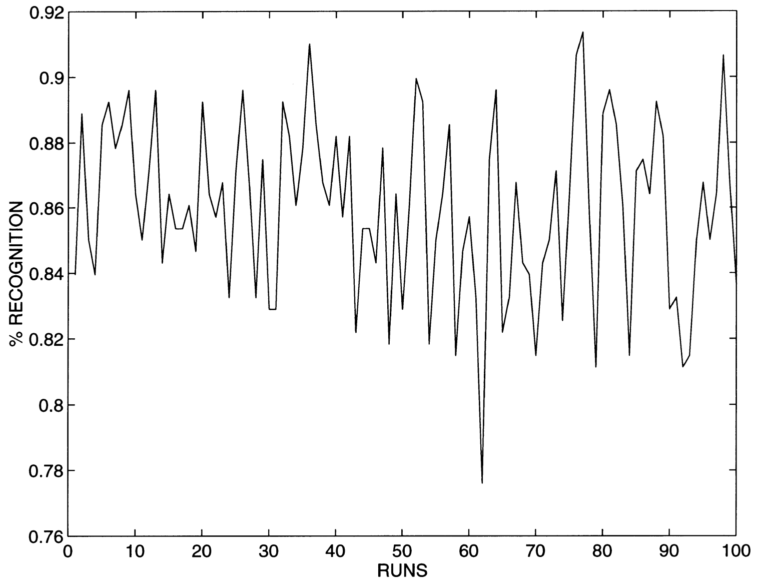


Fig. 5. (a) The number of selected centers and (b) the recognition rate over 100 runs of the algorithm.

mixture models trained with the EM algorithm and some unsupervised networks. We are also exploiting the possibility to use our algorithm for modular neural network design and networks which use different types of hidden units.

In addition, we have started to develop an incremental variant of the algorithm where we start with a small number of training samples and basis functions, and then incrementally add examples and basis functions when needed, while using the selection procedure to remove the redundant ones. Since the method proposed in this paper is very general, it is easy to incorporate all these extensions without changing the general paradigm.

Acknowledgements

This work was supported by grants from the Austrian National Fonds zur Förderung der wissenschaftlichen Forschung (S7002MAT and P10539MAT). A. Leonardis also acknowledges partial support by the Ministry of Science and Technology of Republic of Slovenia (Projects J2-6187, J2-8829), the European Union Copernicus Program (Grant 1068 RECCAD), and by the US–Slovene Joint Board (Project #95-158). We are grateful for the reviewer's constructive comments which helped considerably in improving the quality of the paper.

References

Akaike, H. (1973). Information and an extension of the maximum likelihood principle. In B. N. Petrov & F. Csaki (Eds.), *2nd International Symposion on Information Theory* (pp. 267–281).

Bischof H., & Leonardis A. (1998). Finding optimal neural networks for land use classification. *IEEE Transactions on Geoscience and Remote Sensing, 36 (1)*, 337–341.

Bishop C. M. (1993). Curvature-driven smoothing: a learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks, 4 (5)*, 882–884.

Bishop C. M. (1995). Training with noise is equivalent to Tikhonov regularization. *Neural Computation, 7 (1)*, 108–116.

Chen D. S., & Jain R. C. (1994). A robust back propagation learning algorithm for function approximation. *IEEE Transactions on Neural Networks, 5 (3)*, 467–479.

Cichocki, A. & Unbehauen, R. (1993). *Neural Networks for Optimization and Signal Processing*. New York: Wiley.

Fahlman, S. E. & Lebiere, C. (1990). The cascade-correlation learning architecture. In D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems*, Vol. 2 (pp. 524–532). Morgan Kaufmann.

Fritzke B. (1994). Fast learning with incremental RBF networks. *Neural Processing Letters, 1 (1)*, 2–5.

Fua P. & Hanson A.J. (1989). Objective functions for feature discrimination. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, MI (pp. 1596–1602). Morgan Kaufmann.

Girosi F., Jones M., & Poggio T. (1995). Regularization theory and neural network architectures. *Neural Computation, 7 (2)*, 219–269.

Glover F. & Laguna M., 1993, Tabu search. In C. R. Reeves (Ed.), *Modern Heuristic Techniques for Combinatorial Problems*, (pp. 70–150). Blackwell Scientific Publications.

Hassibi B. & Stork D. G. (1993). Second order derivatives for network pruning: optimal brain surgeon. In S. J. Hanson et al. (Eds.), *NIPS 5*, (pp. 164–172). Morgan Kaufmann.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*. New York: McMillan.

Hinton G. (1989). Connectionist learning procedures. *Artificial Intelligence, 40*, 185–234.

Hochreiter S., & Schmidhuber J. (1997). Flat minima. *Neural Computation, 9 (1)*, 1–43.

Leclerc Y. G. (1989). Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision, 3*, 73–102.

Le Cun, Y.L., Denker, J.S. & Solla S.A. (1988). Optimal brain damage. In D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems 2* (pp. 598–605). San Mateo, CA: Morgan Kaufmann.

Leonardis, A. & Bischof, H. (1996). Dealing with occlusions in the eigenspace approach. In *Proc. of CVPR96* (pp. 453–458). IEEE Computer Society Press.

Leonardis A., Gupta A., & Bajcsy R. (1995). Segmentation of range images as the search for geometric parametric models. *International Journal of Computer Vision, 14 (3)*, 253–277.

Liano K. (1996). Robust error measure for supervised neural network learning with outliers. *IEEE Transactions on Neural Networks, 7 (1)*, 246–250.

Lowe D.G. (1989). Adaptive radial basis function nonlinearities, and the problem of generalization. In *1st IEE Conference on Artificial Neural Networks*, London, UK (pp. 171–175).

MacKay, D. J. C. (1992a) Bayesian interpolation. *Neural Computation, 4*(3), 415–447.

MacKay, D. J. C. (1992b). A practical Bayesian framework for backpropagation networks. *Neural Computation, 4*(3), 448–472.

Moody, J. E. (1992). The effective number of parameters: an analysis of generalization and regularization in nonlinear learning systems. In J. E. Moody, S. J. Hanson & R. P. Lippmann (Eds.), *Advances in Neural Information Processing Systems 4* (pp. 847–854). Morgan Kaufmann.

Moody J. E., & Darken C. (1989). Fast learning in networks of locally tuned processing units. *Neural Computation, 1 (2)*, 281–294.

Mozer M. C. & Smolensky P. (1989). Skeletonization: a technique for trimming the fat from a network via relevance assessment. In D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems*, Vol. 1 (pp. 107–115). San Mateo, CA: Morgan Kaufmann.

Nowlan S. J., & Hinton G. E. (1992). Simplifying neural networks by soft weight sharing. *Neural Computation, 4 (4)*, 473–493.

Orr M. J. (1995). Regularization in the selection of basis function centers. *Neural Computation, 7 (3)*, 606–623.

Pentland A. P. (1990). Automatic extraction of deformable part models. *International Journal of Computer Vision, 4*, 107–126.

Platt C. J. (1991). A resource allocating network for function interpolation. *Neural Computation, 3 (2)*, 213–225.

Poggio T., & Girosi F. (1990). Networks for approximation and learning. *Proceedings of the IEEE, 78 (9)*, 1481–1497.

Rissanen J. (1983). A universal prior for the integers and estimation by minimum description length. *Annals of Statistics, 11 (2)*, 416–431.

Rissanen J. (1984). Universal coding, information, prediction, and estimation. *IEEE Transactions on Information Theory, 30*, 629–636.

Rissanen, J. (1989). *Stochastic Complexity in Statistical Inquiry, Series in Computer Science*, Vol. 15. World Scientific.

Roy A., Govil S., & Miranda R. (1995). An algorithm to generate radial basis function (RBF)-like nets for classification problems. *Neural Networks, 8 (2)*, 179–201.

Sardo L. & Kittler J. (1996). Complexity analysis of RBF networks for pattern recognition. In *Proceedings of the CVPR'96* (pp. 574–579). IEEE Computer Society Press.

Shannon C. (1948). A mathematical theory of communication. *Bell Systems Technical Journal, 27*, 379–423.

Stone M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B:, 36 (1)*, 111–147.

Stricker, M. & Leonardis, A. (1995). ExSel + + : a general framework to extract parametric models. In V. Hlavac & R. Sara (Eds.), *6th CAIP'95, number 970 in Lecture Notes in Computer Science* (pp. 90–97), Prague, Czech Republic, 1995. Springer.

Vapnik V. N. (1982). *Estimation of Dependences Based on Empirical Data*. Springer Verlag.

Weigend A. S., Huberman B. A., & Rumelhart D. E. (1990). Predicting the future: a connectionist approach. *International Journal of Neural Systems, 1 (3)*, 193–209.

Whitehead B. A., & Choate T. D. (1995). Cooperative–competitive genetic evolution of radial basis function centers and widths for time series prediction. *IEEE Transactions on Neural Networks, 5 (1)*, 15–23.

Yingwei L., Sundarajan N., & Saratchandran P. (1997). A sequential learning scheme for function approximation using minimal radial basis function neural networks. *Neural Computation, 9 (2)*, 461–478.

Zemel R. S., & Hinton G. E. (1995). Learning population codes by minimum description length. *Neural Computation, 7 (3)*, 549–564