PERGAMON

Neural Networks 14 (2001) 439–458

Neural Networks

www.elsevier.com/locate/neunet

Contributed article

Three learning phases for radial-basis-function networks

Friedhelm Schwenker*, Hans A. Kestler, Günther Palm

Department of Neural Information Processing, University of Ulm, D-89069 Ulm, Germany

Received 18 December 2000; accepted 18 December 2000

Abstract

In this paper, learning algorithms for radial basis function (RBF) networks are discussed. Whereas multilayer perceptrons (MLP) are typically trained with backpropagation algorithms, starting the training procedure with a random initialization of the MLP's parameters, an RBF network may be trained in many different ways. We categorize these RBF training methods into one-, two-, and three-phase learning schemes.

Two-phase RBF learning is a very common learning scheme. The two layers of an RBF network are learnt separately; first the RBF layer is trained, including the adaptation of centers and scaling parameters, and then the weights of the output layer are adapted. RBF centers may be trained by clustering, vector quantization and classification tree algorithms, and the output layer by supervised learning (through gradient descent or pseudo inverse solution). Results from numerical experiments of RBF classifiers trained by two-phase learning are presented in three completely different pattern recognition applications: (a) the classification of 3D visual objects; (b) the recognition hand-written digits (2D objects); and (c) the categorization of high-resolution electrocardiograms given as a time series (1D objects) and as a set of features extracted from these time series. In these applications, it can be observed that the performance of RBF classifiers trained with two-phase learning can be improved through a third backpropagation-like training phase of the RBF network, adapting the whole set of parameters (RBF centers, scaling parameters, and output layer weights) simultaneously. This, we call three-phase learning in RBF networks. A practical advantage of two- and three-phase learning in RBF networks is the possibility to use unlabeled training data for the first training phase.

Support vector (SV) learning in RBF networks is a different learning approach. SV learning can be considered, in this context of learning, as a special type of one-phase learning, where only the output layer weights of the RBF network are calculated, and the RBF centers are restricted to be a subset of the training data.

Numerical experiments with several classifier schemes including k-nearest-neighbor, learning vector quantization and RBF classifiers trained through two-phase, three-phase and support vector learning are given. The performance of the RBF classifiers trained through SV learning and three-phase learning are superior to the results of two-phase learning, but SV learning often leads to complex network structures, since the number of support vectors is not a small fraction of the total number of data points.

Keywords: Radial basis functions; Initialization and learning in artificial neural networks; Support vector learning; Clustering and vector quantization; Decision trees; Optical character recognition; 3D object recognition; Classification of electrocardiograms

1. Introduction

Radial basis function (RBF) networks were introduced into the neural network literature by Broomhead and Lowe (1988). The RBF network model is motivated by the locally tuned response observed in biologic neurons. Neurons with a locally tuned response characteristic can be found in several parts of the nervous system, for example cells in the auditory system selective to small bands of frequencies (Ghitza, 1991; Rabiner & Juang, 1993) or cells in the visual cortex sensitive to bars oriented in a certain direction or other visual features within a small region of the visual field (see Poggio & Girosi, 1990b). These locally tuned neurons show response characteristics bounded to a small range of the input space.

The theoretical basis of the RBF approach lies in the field of interpolation of multivariate functions. Here, multivariate functions $f : \mathbb{R}^d \rightarrow \mathbb{R}^m$ are considered. We assume that m is equal to 1 without any loss of generality. The goal of interpolating a set of tupels $(\mathbf{x}^\mu, y^\mu)_{\mu=1}^M$ with $\mathbf{x}^\mu \in \mathbb{R}^d$ and $y^\mu \in \mathbb{R}$ is to find a function $F : \mathbb{R}^d \rightarrow \mathbb{R}$ with $F(\mathbf{x}^\mu) = y^\mu$ for all $\mu = 1, \ldots, M$, where F is an element of a predefined set of

* Corresponding author. Tel.: +49-731-50-24159; fax: +49-731-50-24156.

E-mail addresses: schwenker@informatik.uni-ulm.de (F. Schwenker), kestler@informatik.uni-ulm.de (H.A. Kestler), palm@informatik.uni-ulm.de (G. Palm).

0893-6080/01/$ - see front matter
PII: S0893-6080(01)00027-2

functions $\mathcal{F}$, typically $\mathcal{F}$ is a linear space. In the RBF approach the interpolating function F is a linear combination of basis functions:

$$F(\mathbf{x}) = \sum_{\mu=1}^{M} w_\mu h(\|\mathbf{x} - \mathbf{x}^\mu\|) + p(\mathbf{x}) \tag{1}$$

where $\|\cdot\|$ denotes the Euclidean norm, $w_1, \ldots, w_M$ are real numbers, h a real valued function, and p a polynomial $p \in \Pi_n^d$ (polynomials of degree at most n in d variables). The degree of the polynomial term has to be fixed in advance. The interpolation problem is to determine the real coefficients $w_1, \ldots, w_M$ and the polynomial term $p := \sum_{l=1}^{D} a_l p_j$ where $p_1, \ldots, p_D$ is the standard basis of Π_n^d and $a_1, \ldots, a_D$ are real coefficients. The function F has to satisfy the conditions:

$$F(\mathbf{x}^\mu) = y^\mu, \qquad \mu = 1, \ldots, M \tag{2}$$

and

$$\sum_{\mu=1}^{M} w_\mu p_j(\mathbf{x}^\mu) = 0, \qquad j = 1, \ldots, D. \tag{3}$$

Sufficient conditions for the unique solvability of the inerpolation problem were given by several authors (Light, 1992; Micchelli, 1986; Powell, 1992). The function h is called a *radial basis function* if the interpolation problem has a unique solution for any choice of data points. In some cases, the polynomial term in formula (1) can be omitted, and then the interpolation problem is equivalent to the matrix equation

$$\mathbf{H}\mathbf{w} = \mathbf{y} \tag{4}$$

where $\mathbf{w} = (w_1, \ldots, w_M)$, $\mathbf{y} = (y^1, \ldots, y^M)$, and $\mathbf{H}$ is an $M \times M$ matrix defined by

$$\mathbf{H} = \left(h(\|\mathbf{x}^\nu - \mathbf{x}^\mu\|)\right)_{\mu,\nu=1,\ldots,M}. \tag{5}$$

Provided the inverse of $\mathbf{H}$ exists, the solution $\mathbf{w}$ of the interpolation problem can be explicitly calculated and has the form:

$$\mathbf{w} = \mathbf{H}^{-1}\mathbf{y}. \tag{6}$$

Examples of radial basis functions h often used in applications are:

$$h(r) = \mathrm{e}^{-r^2/2\sigma^2} \tag{7}$$

$$h(r) = (r^2 + \sigma^2)^{1/2} \tag{8}$$

$$h(r) = (r^2 + \sigma^2)^{-1/2} \tag{9}$$

Here, σ is a positive real number which we call the *scaling parameter* or the *width* of the radial basis functions. The most popular and widely used radial basis function is the *Gaussian basis function*

$$h(\|\mathbf{x} - \mathbf{c}\|) = \exp(-\|\mathbf{x} - \mathbf{c}\|^2/2\sigma^2) \tag{10}$$

with peak at center $\mathbf{c} \in \mathbb{R}^d$ and decreasing as the distance from the center increases. Throughout this paper we restrict ourselves to this type of radial basis function.

The solution of the exact interpolating RBF mapping passes through every data point $(\mathbf{x}^\mu, y^\mu)$. In the presence of noise, the exact solution of the interpolation problem is typically a function oscillating between the given data points. An additional problem with the exact interpolation procedure is that the number of basis functions is equal to the number of data points and so calculating the inverse of the $M \times M$ matrix $\mathbf{H}$ becomes intractable in practice.

In applications where one has to deal with many thousands of noisy data points, an approximative solution to the data is more desirable than an interpolative one. Broomhead and Lowe (1988) proposed to reduce the number of basis functions in order to reduce the computational complexity. This technique produces a solution by approximating instead of interpolating the data points. Furthermore, in Broomhead and Lowe (1988) an interpretation of the RBF method as an artificial neural network model is given. It consists of three neural layers: a layer of input neurons feeding the feature vectors into the network; a hidden layer of RBF neurons, calculating the outcome of the basis functions; and a layer of output neurons, calculating a linear combination of the basis functions. Under some additional conditions imposed on the basis function h, the set of RBF networks with free adjustable prototype vectors are shown to be universal approximators, so that any continuous function can be approximated with arbitrary precision (Park & Sandberg, 1993). This implies that RBF networks with adjustable prototypes can also be used for classification tasks (Poggio & Girosi, 1990a).

In the classification scenario, the RBF network has to perform a mapping from a continuous input space $\mathbb{R}^d$ into a finite set of classes $Y = \{1, \ldots, L\}$, where L is the number of classes. In the training phase, the parameters of the network are determined from a finite training set

$$S = \{(\mathbf{x}^\mu, y^\mu) | \mathbf{x}^\mu \in \mathbb{R}^d, \qquad y^\mu \in Y, \qquad \mu = 1, \ldots, M\}, \tag{11}$$

here each feature vector $\mathbf{x}^\mu$ is labeled with its class membership y^μ. In the recall phase, further unlabeled observations $\mathbf{x} \in \mathbb{R}^d$ are presented to the network which estimates their class memberships $y \in Y$. In our classification scenario utilizing RBF networks, the number of output units corresponds to the number of classes, and the classmemberships $y \in Y$ are encoded through a 1-of-L coding into a binary vector $\mathbf{z} \in \{0, 1\}^L$ through the relation $\mathbf{z}_i^\mu = 1$ iff $y^\mu = i$. To simplify the notation, we do not distinguish between these two representations of the classmembership. In this context it should be mentioned that other coding schemes can be used but are not very common in pattern recognition applications. Using the 1-of-L encoding scheme an RBF network with K basis functions is performing a mapping $F : \mathbb{R}^d \to \mathbb{R}^L$,

$$F_i(\mathbf{x}) = \sum_{j=1}^{K} w_{ji} h(\|\mathbf{x} - \mathbf{c}_j\|) + w_{0i}, \qquad i = 1, \ldots, L, \tag{12}$$

where the w_{0i} denote the biases, which may be absorbed into the summation by including an extra basis function $h_0 = 1$ whose activation is set equal to 1 on the whole input space $\mathbb{R}^d$. Categorization is performed by assigning the input vector $\mathbf{x}$ the class of the output unit with maximum activation:

$$\text{class}(\mathbf{x}) = \underset{i\in\{1,\ldots,L\}}{\text{argmax}}\, F_i(\mathbf{x}). \tag{13}$$

Typically, an RBF as a neural network model differs from the RBF as an interpolation method in some ways.

1. The number of basis functions is typically much less than the number of data points, and the basis functions are located in representative prototypes $\mathbf{c}_j \in \mathbb{R}^d$ which are not restricted to be data points.
2. Instead of a global scaling parameter $\sigma \in \mathbb{R}$ for all basis functions, each basis function has its own scaling parameter $\sigma_j \in \mathbb{R}$.
3. In some RBF network models the so-called Mahalanobis distance is used instead of the Euclidean distance. In general a Mahalanobis distance between two points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ is defined by a positive definite matrix $\mathbf{R}$ and is given through

$$\|\mathbf{x} - \mathbf{y}\|_{\mathbf{R}} = \sqrt{(\mathbf{x} - \mathbf{y})^T \mathbf{R} (\mathbf{x} - \mathbf{y})} \tag{14}$$

here T denotes the transpose of a matrix. Typically $\mathbf{R}$ is the inverse of the covariance matrix of the input data points $\mathbf{x}^\mu$, $\mu = 1, \ldots, M$. The Mahalanobis distance becomes the Euclidean distance if $\mathbf{R}$ is equal to the identity matrix $\mathbf{I}$.

In this type of RBF network, every basis function has its own matrix $\mathbf{R}_j$, usually defined as the inverse of the covariance matrix of the data points with respect to the center $\mathbf{c}_j$. Such an architecture contains d parameters for each center $\mathbf{c}_j$ plus $d(d+1)/2$ parameters for each matrix $\mathbf{R}_j$. In some approaches, the matrix $\mathbf{R}_j$ is simplified to be a diagonal matrix for every center.

To simplify the notation, we set $h_j(\mathbf{x}) = h(\|\mathbf{x} - \mathbf{c}_j\|^2_{\mathbf{R}_j})$, $j = 1, \ldots, K$ and the RBF network (12) becomes

$$F_i(\mathbf{x}) = \sum_{j=0}^{K} w_{ji} h_j(\mathbf{x}), \qquad i = 1, \ldots, L. \tag{15}$$

With these modifications, the process of adjusting the parameters is usually treated as a typical neural network training process. In many applications, the first (RBF) layer and the second (combination weight) layer are trained separately. This has led to a bad reputation of RBF networks in some application areas, which is due to the impression that the performance of RBF networks after these two training phases is worse than, for example, that of multilayer perceptrons (MLP) networks (Michie, Spiegelhalter, & Taylor, 1994). However, a combined training of the whole network in the style of backpropagation has also been proposed (Poggio & Girosi, 1990a) which leads to a better performance comparable to MLP networks. Here, we advocate a training procedure in three phases that combines these approaches.

We distinguish the following three learning or training schemes, which can be used for RBF networks.

One-phase learning. With this learning procedure, only the output layer weights $\mathbf{w}$ are adjusted through some kind of supervised optimization, e.g. minimizing the squared difference between the network's output and the desired output value. Here, the centers $\mathbf{c}_j$ are sub-sampled from the set of input vectors $\mathbf{x}^\mu$ (or all data points are used as centers) and, typically, all scaling parameters are set equal to a predefined real number σ.

Support vector learning is a special example of one-phase learning. Here, only the output layer weights are adjusted, the location of the kernel centers is restricted to the data points $\{\mathbf{x}^\mu \in \mathbb{R}^d : \mu = 1, \ldots, M\}$ and the scaling parameter is fixed in advance (see Appendix A).

Two-phase learning. Here, the two layers of the RBF network are trained separately, first RBF centers $\mathbf{c}_j$ and the scaling parameters are determined, and subsequently the output layer is adjusted (see Section 2).

Three-phase learning. After the initialization of the RBF networks utilizing two-phase learning, the whole architecture is adjusted through a further optimization procedure (see Section 3).

The paper is organized in the following way. In Section 2 we introduce the classical two-stage training of the two layers. Backpropagation learning for RBF networks is reviewed in Section 3. In Section 4 a brief description of the different classifiers used in the evaluation is given and we demonstrate the superiority of three-stage training in different application domains: (a) the classification of 3D visual objects; (b) the recognition of hand-written digits (2D objects); and (c) the categorization of high-resolution electrocardiograms given as a time series (1D objects) and as a set of features extracted from these time series. We end with some conclusions in Section 5. Support vector learning as a special type of one-phase learning scheme for RBF networks is reviewed in Appendix A.

2. Two-phase learning for RBF networks

In a multilayer perceptron (MLP) network all parameters are usually adapted simultaneously by an optimization procedure. This training procedure is supervised, since it minimizes an error function measuring the difference between the network output and the teacher signal that provides the correct output. In contrast to training an MLP network, learning in an RBF network can be done in two stages.

1. Adjusting the parameters of the RBF layer, including the RBF centers $\mathbf{c}_j \in \mathbb{R}^d, j = 1, \ldots, K$, and the scaling parameters given through scalars, vectors, or matrices $\mathbf{R}_j \in$

$\mathbb{R}, \in \mathbb{R}^d$, or $\mathbb{R}^{d^2}$, respectively. Here, d is the dimension of the input space.

2. Calculation of the output weights $\mathbf{w}_j \in \mathbb{R}^L$ for $j = 1, \ldots, L$ of the network. L is the number of classes.

To determine the centers for the RBF networks, typically unsupervised training procedures from clustering are used (Moody & Darken, 1989), whereas in the original use for interpolation the centers are simply the data points. If the RBF network has to perform a classification task, supervised training procedures to determine the RBF centers are also applicable, because the target values of the input vectors are given.

In this section we present three completely different types of algorithms to initialize the RBF centers: unsupervised clustering methods (Moody & Darken, 1989), supervised vector quantization (Schwenker, Kestler, Palm, & Höher, 1994), and supervised training of decision trees (Kubat, 1998; Schwenker & Dietrich, 2000). We then describe heuristics to calculate the scaling parameters of the basis functions and discuss supervised training methods for the output layer weights.

2.1. Vector quantization to calculate the RBF centers

Clustering and vector quantization techniques are typically used when the data points have to be divided into natural groups and no teacher signal is available. Here, the aim is to determine a small but representative set of centers or prototypes from a larger data set in order to minimize some quantization error. In the classification scenario where the target classification of each input pattern is known, supervised vector quantization algorithms, such as Kohonen's learning vector quantization (LVQ) algorithm, can also be used to determine the prototypes. We briefly describe k-means clustering and LVQ learning in the following.

2.1.1. Unsupervised competitive learning

A competitive neural network consists of a single layer of K neurons. Their synaptic weights vectors $\mathbf{c}_1, \ldots, \mathbf{c}_K \in \mathbb{R}^d$ divide the input space into K disjoint regions $\mathscr{R}_1, \ldots, \mathscr{R}_K \subset \mathbb{R}^n$, where each set $\mathscr{R}_j$ is defined by

$$\mathscr{R}_j = \{\mathbf{x} \in \mathbb{R}^d | \, \|\mathbf{x} - \mathbf{c}_j\| = \min_{i=1,\ldots,K} \|\mathbf{x} - \mathbf{c}_j\|\}. \tag{16}$$

Such a partition of the input space is called a Voronoi tessellation where each weight vector $\mathbf{c}_j$ is a representative prototype vector for region $\mathscr{R}_j$.

When an input vector $\mathbf{x} \in \mathbb{R}^n$ is presented to the network, all units $j = 1, \ldots, k$ determine their Euclidean distance to $\mathbf{x}$: $d_j = \|\mathbf{x} - \mathbf{c}_j\|$.

Competition between the units is realized by searching for the minimum distance $d_{j^*} = \min_{j=1,\ldots,K} d_j$. The corresponding unit with index j^* is called the winner of the competition and this winning unit is trained through the *unsupervised competitive learning* rule

$$\Delta \mathbf{c}_{j^*} = \eta_t (\mathbf{x}^\mu - \mathbf{c}_{j^*}) \tag{17}$$

where $\mathbf{c}_{j^*}$ is the closest prototype to the input $\mathbf{x}^\mu$. For convergence, the learning rate η_t has to be a sequence of positive real numbers such that $\eta_t \to 0$ as the number of data points presentations t grows up to ∞, $\sum_{t=1}^{\infty} \eta_t = \infty$ and $\sum_{t=1}^{\infty} \eta_t^2 < \infty$.

One of the most popular methods in the cluster analysis is the k-means clustering algorithm. The empirical quantization error defined by

$$E(\mathbf{c}_1, \ldots, \mathbf{c}_K) = \sum_{j=1}^{K} \sum_{\mathbf{x}^\mu \in \mathscr{C}_j} \|\mathbf{x}^\mu - \mathbf{c}_j\|^2, \tag{18}$$

is minimal, if each prototype $\mathbf{c}_j$ is equal to the corresponding center of gravity of data points $\mathscr{C}_j := \mathscr{R}_j \cap \{\mathbf{x}^1, \ldots, \mathbf{x}^M\}$. Starting from a set of initial seed prototypes, these are adapted through the learning rule

$$\mathbf{c}_j = \frac{1}{|\mathscr{C}_j|} \sum_{\mathbf{x}^\mu \in \mathscr{C}_j} \mathbf{x}^\mu, \tag{19}$$

which is called *batch mode k-means clustering*. The iteration process can be stopped if the sets of data points within each cluster $\mathscr{C}_j$ in two consecutive learning epochs are equal. Incremental optimization of E can also be realized utilizing learning rule (17) or

$$\Delta \mathbf{c}_j * = \frac{1}{N_{j^*} + 1} (\mathbf{x}^\mu - \mathbf{c}_{j^*}) \tag{20}$$

N_{j^*} counts how often unit j^* was the winning unit of the competition. The topic of incremental clustering algorithm has been discussed by Darken and Moody (1990).

Prototypes $\mathbf{c}_1, \ldots, \mathbf{c}_K$ trained through bate mode k-means, incremental k-means, or the general unsupervised competitive learning rule can serve as initial locations of centers of the basis functions in RBF networks.

2.1.2. LVQ learning

It is assumed that a classification or pattern recognition task has to be performed by the RBF network. A training set of feature vectors $\mathbf{x}^\mu$ is given each labeled with a target classification y^μ. In this case, supervised learning may be used to determine the set of prototype vectors $\mathbf{c}_1, \ldots, \mathbf{c}_K$.

The LVQ learning algorithm has been suggested by Kohonen (1990) for vector quantization and classification tasks. From the basic LVQ1 version, LVQ2, LVQ3 and OLVQ1 training procedures have been derived. OLVQ1 denotes the optimized LVQ algorithm. Presenting a vector $\mathbf{x}^\mu \in \mathbb{R}^d$ together with its classmembership the winning prototype j^* is adapted according to the LVQ1-learning rule:

$$\Delta \mathbf{c}_{j^*} = \eta_t (\mathbf{x}^\mu - \mathbf{c}_{j^*}) \left(y_{j^*}^\mu - \frac{1}{2} z_{j^*}^\mu \right). \tag{21}$$

Here, $\mathbf{z}^\mu$ is the binary output of the network and $\mathbf{y}^\mu$ is a

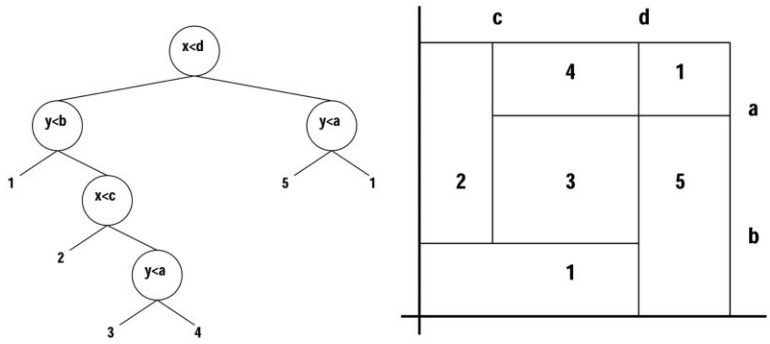

Fig. 1. A binary decision tree of depth 4 with two features, denoted by x and y, is given (left panel). The data stem from five different classes (denoted by 1, 2, 3, 4, 5) of a two dimensional feature space. Each node is labeled with the selected feature and a boundary value. The corresponding partition into hyperrectangles parallel to the axes of the feature space is shown (right figure), here boundary values and class labels are given. The minima and maxima of each feature within the training set are additional boundary values. Thus, all regions are bounded.

binary target vector coding the classmembership for feature input vector $\mathbf{x}^\mu$. In both vectors $\mathbf{z}^\mu$ and $\mathbf{y}^\mu$, exactly one component is equal to 1, all others are 0. The difference $s_{j^*}^\mu = 2(y_{j^*}^\mu - z_{j^*}^\mu/2)$ is equal to 1 if the classification of the input vector is correct and -1 if it is a false classification by the class label of the nearest prototype. In the LVQ1, LVQ2 and LVQ3 algorithms, η_t is a positive decreasing learning rate. For the OLVQ1 algorithm, the learning rate depends on the actual classification by the winning prototype, and is not decreasing in general. It is defined by

$$\eta_t = \frac{\eta_t}{1 + s_{j^*}^\mu \eta_t}. \tag{22}$$

For a detailed treatment on LVQ learning algorithms see Kohonen (1995). After LVQ training, the prototypes $\mathbf{c}_1$, …, $\mathbf{c}_K$ can be used as the initial RBF centers (Schwenker et al., 1994).

2.2. Decision trees to calculate the RBF centers

Decision trees (or classification trees) divide the feature space $\mathbb{R}^d$ into pairwise disjoint regions $\mathscr{R}_j$. The binary decision tree is the most popular type. Here, each node has either two or zero children. Each node in a decision tree represents a certain region $\mathscr{R}$ of $\mathbb{R}^d$. If the node is a terminal node, called a leaf, all data points within this region $\mathscr{R}$ are classified to a certain class. If a node has two children then the two regions represented by the children nodes, denoted by $\mathscr{R}_{\text{left}}$ and $\mathscr{R}_{\text{right}}$ form a partition of $\mathscr{R}$, i.e. $\mathscr{R}_{\text{left}} \cup \mathscr{R}_{\text{right}} = \mathscr{R}$ and $\mathscr{R}_{\text{left}} \cap \mathscr{R}_{\text{right}} = \emptyset$. Typical decision tree algorithms calculate a partition with hyperrectangles parallel to the axes of the feature space, see Fig. 1.

Kubat (1998) presents a method to transform such a set of disjoint hyperrectangular regions $\mathscr{R}_j \subset \mathbb{R}^d$, represented by the leaves of the decision tree, into a set of centers $\mathbf{c}_j \in \mathbb{R}^d$ and scaling parameters in order to initialize a Gaussian basis function network. Many software packages are available to calculate this type of binary decision tree. In the numerical experiments given in this paper, Quinlan's C4.5 software was used (Quinlan, 1992).

In Fig. 1, a decision tree and the set of regions, defined through the tree's leaves are shown. Each terminal node of the decision tree determines a rectangular region in the feature space $\mathbb{R}^d$, here $d = 2$. In the binary classification tree, each node is determined by a feature dimension $i \in \{1, \ldots, d\}$ and a boundary $b_i \in \mathbb{R}$. For each of the features, the minimum and maximum are additional boundary values, see Fig. 1. Typically, the data points of a single class are located in different parts of the input space, and thus a class is represented by more than one leaf of the decision tree. For instance, class 1 is represented by two leaves in Fig. 1. Each region $\mathscr{R}$, represented by a leaf, is completely defined by a path through the tree starting at the root and terminating in a leaf.

For each region $\mathscr{R}_j$, represented by a leaf of the decision tree, with

$$\mathscr{R}_j = [a_{1j}, b_{1j}] \times \cdots \times [a_{dj}, b_{dj}] \tag{23}$$

an RBF center $\mathbf{c}_j = (c_{1j}, \ldots, c_{dj})$ is determined through

$$c_{ij} = (a_{ij} + b_{ij})/2, \qquad i = 1, \ldots, d. \tag{24}$$

2.3. Calculating the kernel widths

The setting of the kernel widths is a critical issue in the transition to the RBF network (Bishop, 1995). When the kernel width $\sigma \in \mathbb{R}$ is too large, the estimated probability density is over-smoothed and the nature of the underlying true density may be lost. Conversely, when σ is too small there may be an over-adaptation to the particular data set. In addition, very small or large σ tend to cause numerical problems with gradient descent methods as their gradients vanish.

In general the Gaussian basis functions $h_1, \ldots, h_K$ have the form

$$h_j(\mathbf{x}) = \exp(-(\mathbf{x} - \mathbf{c}_j)^T \mathbf{R}_j (\mathbf{x} - \mathbf{c}_j)) \tag{25}$$

where each $\mathbf{R}_j$, $j = 1, \ldots, K$, is a positive definite $d \times d$ matrix. Girosi, Jones, & Poggio (1995) called this type of basis function a hyper basis function. The contour of a basis

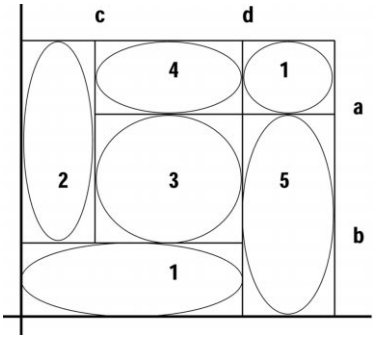

Fig. 2. The regions in the feature space defined through the leaves of the decision tree. Centers of the RBFs are located in the middle of each hyperrectangle. The contour of the radial basis functions $h_j(x)$ are hyperellipsoids.

function, more formally the set $H_j^\alpha = \{\mathbf{x} \in \mathbb{R}^d | h_j(\mathbf{x}) = \alpha\}$, $\alpha > 0$, is a hyperellipsoid in $\mathbb{R}^d$, see Fig. 2. Depending on the structure of the matrices $\mathbf{R}_j$, four types of hyperellipsoids appear.

1. $\mathbf{R}_j = 1/2\sigma^2\mathbf{Id}$ with $\sigma^2 > 0$. In this case all basis functions h_j have a radial symmetric contour all with constant width, and the Mahalanobis distances reduces to the Euclidean distance multiplied by a fixed constant scaling parameter. This is the original setting of RBF in the context of interpolation and support vector machines.
2. $\mathbf{R}_j = 1/2\sigma_j^2\mathbf{Id}$ with $\sigma_j^2 > 0$. Here the basis functions are radially symmetric, but are scaled with different widths.
3. $\mathbf{R}_j$ are diagonal matrices, but the elements of the diagonal are not constant. Here, the contour of a basis function h_j is not radially symmetric—in other words the axes of the hyperellipsoids are parallel to the axes of the feature space, but of different length, see Fig. 2.
   In this case $\mathbf{R}_j$ is completely defined by a d-dimensional vector $\sigma_j \in \mathbb{R}^d$:
$$\mathbf{R}_j = \mathbf{Id}\left(\frac{1}{2\sigma_{1j}^2}, \ldots, \frac{1}{2\sigma_{dj}^2}\right) = \mathrm{diag}\left(\frac{1}{2\sigma_{1j}^2}, \ldots, \frac{1}{2\sigma_{dj}^2}\right). \quad (26)$$
4. $\mathbf{R}_j$ is positive definite, but not a diagonal matrix. This implies that shape and orientation of the axes of the hyperellipsoids are arbitrary in the feature space.

We investigated different schemes for the initial setting of the real-valued and vector-valued kernel widths in transition to the RBF network. In all cases, a parameter $\alpha > 0$ has to be set heuristically.

1. All σ_j are set to the same value σ, which is proportional to the average of the p minimal distances between all pairs of prototypes. First, all distances $d_{lk} = \|\mathbf{c}_l - \mathbf{c}_k\|$ with $l = 1, \ldots, K$ and $k = l + 1, \ldots, K$ are calculated and then renumbered through an index mapping $(l, k) \rightarrow (l-1)K + (k-1)$. Thus, there is a permutation τ such that the distances are arranged as an increasing sequence with $d_{\tau(1)} \leq d_{\tau(2)} \leq \cdots \leq d_{\tau(K(K-1)/2)}$ and $\sigma_j = \sigma$ is set to:
$$\sigma_j = \sigma = \alpha\frac{1}{p}\sum_{i=1}^{p} d_{\tau(i)}. \quad (27)$$
2. The kernel width σ_j is set to the mean of the distance to the p nearest prototypes of $\mathbf{c}_j$. All distances $d_{lj} = \|\mathbf{c}_l - \mathbf{c}_j\|$ with $l = 1, \ldots, K$ and $l \neq j$ are calculated and renumbered through a mapping $(l, j) \rightarrow l$ for $l < j$ and $(l, j) \rightarrow l - 1$ for $l > j$, then there is a permutation τ such that $d_{\tau(1)} \leq d_{\tau(2)} \leq \cdots \leq d_{\tau(K-1)}$ and σ_j is set to:
$$\sigma_j = \alpha\frac{1}{p}\sum_{i=1}^{p} d_{\tau(i)} \quad (28)$$
3. The distance to the nearest prototype with a different class label is used for the initialization of σ_j:
$$\sigma_j = \alpha\min\{\|\mathbf{c}_i - \mathbf{c}_j\| : class(\mathbf{c}_i) \neq class(\mathbf{c}_j), \quad i = 1, \ldots, K\} \quad (29)$$
4. The kernel width σ_j is set to the mean of distances between the data points of the corresponding cluster C_j:
$$\sigma_j = \alpha\frac{1}{|\mathscr{C}_j|}\sum_{\mathbf{x}^\mu \in \mathscr{C}_j} \|\mathbf{x}^\mu - \mathbf{c}_j\| \quad (30)$$

In the situation of vector-valued kernel parameters, the widths $\sigma_j \in \mathbb{R}^d$ may be initially set to the variance of each input feature based on all data points in the corresponding cluster $\mathscr{C}_j$:
$$\sigma_{ij}^2 = \alpha\frac{1}{|\mathscr{C}_j|}\sum_{\mathbf{x}^\mu \in \mathscr{C}_j} (\mathbf{x}_i^\mu - \mathbf{c}_{ij})^2 \quad (31)$$

In the case of RBF network initialization using decision trees, the kernel parameters can be defined through the sizes of the regions $\mathscr{R}_j$. In this case, the kernel widths are given by a diagonal matrix $\mathbf{R}_j$, which is determined through a vector $\sigma_j \in \mathbb{R}^d$. The size of the hyperrectangle $\mathscr{R}_j$ defines the shape of a hyperellipsoid:
$$\sigma_j = \frac{\alpha}{2}((b_{1j} - a_{1j}), \ldots, (b_{dj} - a_{dj})). \quad (32)$$

These widths are determined in such a way that all RBFs have the same value at the border of their corresponding region (see Fig. 2). Kubat (1998) proposed a slightly different method, where the RBF centers c_j are placed in the middle of the region $\mathscr{R}_j$, except that the region touches the border of an input feature i. In this case, the center $\mathbf{c}_j$ is placed at this border and the scaling parameter σ_{ij} is multiplied by a factor of two.

In general, the location and the shape of the kernels represented by the centers $\mathbf{c}_j$ and the scaling matrices $\mathbf{R}_j$ can be calculated using a re-estimation technique known as the expectation-maximization (EM) algorithm (Ripley, 1996).

2.4. Training the output weights of the RBF network

Provided that the centers $\mathbf{c}_j$ and the scaling parameters, given by the matrices $\mathbf{R}_j$, of the basis functions have been determined, the weights of the output layer can be calculated (Bishop, 1995; Hertz, Krogh, & Palmer 1991). We assume K basis functions in the hidden layer of the RBF network. Let $(\mathbf{x}^\mu, \mathbf{y}^\mu)$, $\mu = 1, \ldots, M$ be the set of training examples with feature vector $\mathbf{x}^\mu \in \mathbb{R}^d$ and target $\mathbf{y}^\mu \in \mathbb{R}^m$, $H_{\mu j} = h_j(\mathbf{x}^\mu)$ the outcome of the j-th basis function with the μ-th feature vector $\mathbf{x}^\mu$ as input, and $Y_{\mu \mathrm{j}}$ the j-th component of the μ-th target vector $\mathbf{y}^\mu$. Given these two matrices $\mathbf{H} = (H_{\mu j})$ and $\mathbf{Y} = (Y_{\mu j})$, the matrix of the output layer weights $\mathbf{W}$ is the result of a minimization of the error function:

$$E(\mathbf{W}) = \|\mathbf{HW} - \mathbf{Y}\|^2. \tag{33}$$

The solution is given explicitly in the form $\mathbf{W} = \mathbf{H}^+\mathbf{Y}$ where $\mathbf{H}^+$ denotes the pseudo inverse matrix of $\mathbf{H}$ which can be defined as

$$\mathbf{H}^+ = \lim_{\alpha \to 0+} (\mathbf{H}^T\mathbf{H} + \alpha \mathbf{Id})^{-1}\mathbf{H}^T. \tag{34}$$

Provided that the inverse matrix of $(\mathbf{H}^T\mathbf{H})$ is defined, the pseudo inverse matrix becomes simply $\mathbf{H}^+ = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$. The solution $\mathbf{W}$ is unique and can also be found by gradient descent optimization of the error function defined in (33). This leads to the delta learning rule for the output weights

$$\Delta w_{jp} = \eta \sum_{\mu=1}^{M} h_j(\mathbf{x}^\mu)(y_p^\mu - F_p(\mathbf{x}^\mu)), \tag{35}$$

or its incremental version

$$\Delta w_{jp} = \eta_t h_j(\mathbf{x}^\mu)(y_p^\mu - F_p(\mathbf{x}^\mu)). \tag{36}$$

After this final step of calculating the output layer weights, all parameters of the RBF network have been determined.

3. Backpropagation and three-phase learning in RBF networks

As described in Section 2 learning in an RBF network can simply be done in two separate learning phases: calculating the RBF layer and then the output layer. This is a very fast training procedure but often leads to RBF classifiers with bad classification performance (Michie et al., 1994). We propose a third training phase of RBF networks in the style of backpropagation learning in MLPs, performing an adaptation of all types of parameters simultaneously. We give a brief summary of the use of error-back-propagation in the context of radial basis function network training (for a more detailed treatment see Bishop, 1995; Hertz et al., 1991; Wasserman, 1993).

If we define as the error function of the network a differentiable function like the sum-of-squares error E,

$$E = \frac{1}{2} \sum_{\mu=1}^{M} \sum_{p=1}^{L} (y_p^\mu - F_p^\mu)^2, \tag{37}$$

with F_p^μ and y_p^μ as the actual and target output values, respectively, and consider a network with differentiable activation functions then a necessary condition for a minimal error is that its derivatives with respect to the parameters center location $\mathbf{c}_j$, kernel width $\mathbf{R}_j$, and output weights $\mathbf{w}_j$ vanish. In the following, we consider the case that $\mathbf{R}_j$ is a diagonal matrix defined by a vector $\sigma_j \in \mathbb{R}^d$.

An iterative procedure for finding a solution to this problem is gradient descent. Here, the full parameter set $\mathbf{U} = (\mathbf{c}_j, \sigma_j, \mathbf{w}_j)$ is moved by a small distance η in the direction in which E decreases most rapidly, i.e. in the direction of the negative gradient $-\nabla E$:

$$\mathbf{U}^{(\tau+1)} = \mathbf{U}^{(\tau)} - \eta \nabla E(\mathbf{U}^{(\tau)}). \tag{38}$$

For the RBF network (15) for the Gaussian basis function, we obtain the following expression rules for the adaptation rules or the network parameters:

$$\Delta w_{jk} = \eta \sum_{\mu=1}^{M} h_j(\mathbf{x}^\mu)(y_k^\mu - F_k^\mu), \tag{39}$$

$$\Delta c_{ij} = \eta \sum_{\mu=1}^{M} h_j(\mathbf{x}^\mu) \frac{x_i^\mu - c_{ij}}{\sigma_{ij}^2} \sum_{p=1}^{L} w_{jp}(y_p^\mu - F_p^\mu), \tag{40}$$

$$\Delta \sigma_{ij} = \eta \sum_{\mu=1}^{M} h_j(\mathbf{x}^\mu) \frac{(x_i^\mu - c_{ij})^2}{\sigma_{ij}^3} \sum_{p=1}^{L} w_{jp}(y_p^\mu - F_p^\mu). \tag{41}$$

Choosing the right learning rate or stepsize η is sometimes a critical issue in neural network training. If its value is too low, convergence to a minimum is slow. Conversely, if it is chosen too high, successive steps in parameter space overshoot the minimum of the error surface. This problem can be avoided by a proper stepwise tuning. A procedure for obtaining such a stepsize was proposed by Armijo (1966). In the following very brief description of the method we draw heavily from the papers of Armijo (1966) and Magoulas, Vrahatis, and Androulakis (1997), for details see the respective articles. Under mild conditions on the error function E, which are satisfied in our setting the following theorem holds:

Theorem (Armijo, 1966). If η_0 is an arbitrarily assigned positive number, $\eta_m = \eta_0/2^{m-1}$, $m \in \mathbb{N}$, then the sequence of weight vectors $(\mathbf{U}^{(\tau)})_{\tau=0}^{\infty}$ defined by

$$\mathbf{U}^{(\tau+1)} = \mathbf{U}^{(\tau)} - \eta_{m_\tau} \nabla E(\mathbf{U}^{(\tau)}), \qquad \tau = 0, 1, 2, \ldots \tag{42}$$

Algorithm 1
Backpropagation with variable stepsize

Require: $E_{\min}, \eta_{\min}, \tau_{\max}$
  $\tau = 0$, $\eta = 1/2$
  while $E(\mathbf{U}^{(\tau)}) > E_{\min} \& \tau \leq \tau_{\max}$ **do**
    if $\tau > 0$ **then**
      $\eta = \|\mathbf{U}^{(\tau)} - \mathbf{U}^{(\tau-1)}\| / 2 \|\nabla E(\mathbf{U}^{(t)}) - \nabla E(\mathbf{U}^{(\tau-1)})\|$
    end if
    while $\eta < \eta_{\min}$ **do**
      $\eta = 2\eta$
    end while
    while $E(\mathbf{U}^{(\tau)} - \eta \nabla E(\mathbf{U}^{(\tau)})) > -\frac{1}{2}\eta \|\nabla E(\mathbf{U}^{(\tau)})\|^2$ **do**
      $\eta = \eta/2$
    end while
    $\mathbf{U}^{(\tau+1)} = \mathbf{U}^{(\tau)} - \eta \nabla E(\mathbf{U}^{(\tau)})$
    $\tau = \tau + 1$
end while

where m_τ is the smallest positive integer for which

$$E(\mathbf{U}^{(\tau)} - \eta_{m_\tau} \nabla E(\mathbf{U}^{(\tau)})) - E(\mathbf{U}^{(\tau)}) \leq -\frac{1}{2} \eta_{m_\tau} \|\nabla E(\mathbf{U}^{(\tau)})\|^2, \tag{43}$$

converges to $\mathbf{U}^*$ which minimizes error function E (locally) starting from the initial vector $\mathbf{U}^{(0)}$.

Using Armijo's theorem Magoulas et al. (1997) proposed a backpropagation algorithm with variable stepsize, see (Algorithm 1).

4. Applications

In the following sections we will compare different methods of initialization and optimization on three different data sets. Support vector (SV) learning results for RBF networks are also given.

Classifiers. For numerical evaluation the following classification schemes were applied.

1NN: Feature vectors are classified through the 1-nearest-neighbor (1NN) rule. Here, the 1NN rule is applied to the whole training set.

LVQ: The 1-nearest-neighbor classifier is trained through Kohonen's supervised OLVQ1 followed by LVQ3 training (each for 50 training epochs). The 1NN rule is applied to the found prototypes (see Section 2.1).

D-Tree: The decision tree is generated by Quinlan's C4.5 algorithm on the training data set (see Section 2.2).

2-Phase-RBF (data points): A set of data points is randomly selected from the training data set. These data points serve as RBF centers. A single scaling parameter per basis function is determined as the mean of the three closest prototypes, see Section 2.3. The weights of the output layer are calculated through the pseudo inverse solution as described in Section 2.4.

2-Phase-RBF (*k*-means): A set of data points is randomly selected from the training data set. These data points are the seeds of an incremental k-means clustering procedure and these k-means centers are used as centers in the RBF network. For each basis function, a single scaling parameter is set to the mean of the three closest prototypes, and the output layer matrix is calculated through the pseudo inverse matrix solution.

2-Phase-RBF (LVQ): A set of data points is randomly selected from the training set. These data points are the seeds for the OLVQ1 training algorithm (50 training epochs), followed by LVQ3 training with again 50 epochs. These prototypes then are used as the centers in the RBF network. A single scaling parameter per basis function is set to the mean of the three closest prototypes and the output layer is calculated through the pseudo inverse matrix.

2-Phase-RBF (D-Tree): The decision tree is trained through Quinlan's C4.5 algorithm. From the resulting decision tree, the RBF centers and the scaling parameters are determined through the transformation described in Section 2.3. Finally, the weights of the output layer are determined through the pseudo inverse matrix.

3-Phase-RBF (data points): The **2-Phase-RBF (data points)** network is trained through a third error-backpropagation training procedure with 100 training epochs (see Section 3).

3-Phase-RBF (*k*-means): The **2-Phase-RBF (*k*-means)** network is trained through a third error-backpropagation training procedure with 100 training epochs.

3-Phase-RBF (LVQ): The **2-Phase-RBF (LVQ)** network is trained through a third error-backpropagation training procedure with 100 training epochs.

3-Phase-RBF (D-Tree): The **2-Phase-RBF (D-Tree)** network is trained through a third error-backpropagation training procedure with 100 training epochs.

SV-RBF: The RBF network with Gaussian kernel function is trained by support vector learning (see Appendix A). For the optimization the NAG library is used. In multi-class applications (number of classes $L > 2$) an RBF network has been trained through SV learning for each class. In the classification phase, the estimate for an unseen exemplar is found through maximum detection among the L classifiers. This is called the *one-against-rest* strategy (Schwenker, 2000).

Evaluation procedure. The classification performance is given in terms of *k-fold cross-validation* results. A k-fold cross-validation means partitioning the whole data set into k disjoint subsets and carrying out k training and test runs always using $k - 1$ subsets as the training set and testing on the remaining one. The results are those on the test sets.

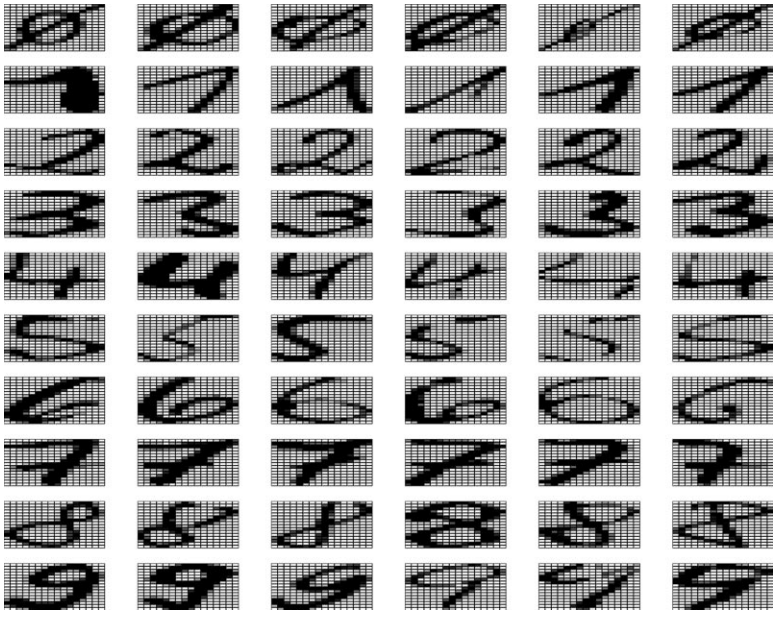

Fig. 3. A subset of 60 hand-written digits with six exemplars of each class sampled from the training data set.

Each of these k-fold cross-validation simulations was performed several times, five times for the 3D object recognition, and ten times in the ECG categorization application. For the evaluation of the hand-written digits, the evaluation was performed on a separate test set. Between subsequent cross-validation runs, the order of the data points was randomly permuted.

4.1. Application to hand-written digits

The classification of machine-printed or hand-written characters is one of the classical applications in the field of pattern recognition and machine learning. In optical character recognition (OCR) the problem is to classify characters into a set of classes (letters, digits, special characters (e.g. mathematical characters), characters from different fonts, characters in different sizes, etc.). After some preprocessing and segmentation, the characters are sampled with a few hundred pixels and then categorized into a class of the predefined set of character categories. In this paper we consider the problem of hand-written digit recognition, which appears as an important subproblem in the area of automatic reading of postal addresses.

4.1.1. Data

The data set used for the evaluation of the performance of the RBF classifiers consist of 20,000 hand-written digits (2000 samples of each class). The digits, normalized in height and width, are represented through a 16×16 matrix G, where the entries $G_{ij} \in \{0, \ldots, 255\}$ are values taken from an 8-bit gray scale, see Fig. 3. Previously, this data set has been used for the evaluation of machine learning techniques in the STATLOG project. Details concerning this data set and the STATLOG project can be found in the final report of STATLOG (Michie et al., 1994).

4.1.2. Results

The whole data set has been divided into a set of 10,000 training samples and a set of 10,000 test samples (1000 examples of each class in both data sets). The training set was used to design the classifiers, and the test set was used for performance evaluation. Three different classifiers per architecture were trained, and the classification error was measured on the test set. For this data set, we present results for all classifier architectures described above. Furthermore, results for multilayer perceptrons **MLP**, and results achieved with the first 40 principal components of this data set for the quadratic polynomial classifier **Poly40**, for the RBF network with SV learning **SV-RBF40**, and for RBF network trained by three-phase RBF learning and LVQ prototype initialization **3-Phase-RBF40 (LVQ)** are given, see Table 1.

For the LVQ classifier, 200 prototypes (20 per class) are used. The RBF networks initialized through randomly selected data points, through centers calculated utilizing k-means clustering or learning vector quantization also consisted of RBF centers. The MLP networks consisted of a single hidden layer with 200 sigmoidal units. The decision tree classifier was trained by Quinlan's C4.5 algorithm. It has been trained with the default parameter settings leading to an RBF network with 505 centers.

Further results for this data set of hand-written digits can also be found in the final STATLOG report. The error rates for the **1NN**, **LVQ**, and **MLP** classifiers are similar in both studies. The error rate for the RBF classifier in Michie et al. (1994) is close to our results achieved by **2-Phase RBF** classifiers with an initialization of the RBF centers utilizing

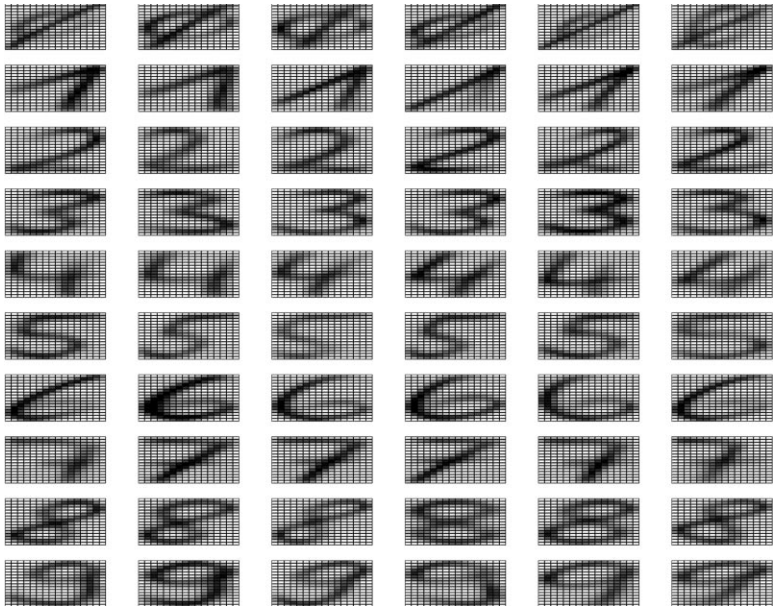

Fig. 4. The 60 cluster centers of the hand-written digits after running the incremental k-means clustering algorithm for each of the 10 digits separately. For each of the 10 digits, $k = 6$ cluster centers are used in this clustering process. The cluster centers are initialized through data points that are randomly selected from the training data set.

k-means, **LVQ**, and D-Tree. Indeed, the RBF classifiers considered in Michie et al. (1994) were trained in two separate stages. First, the RBF centers were calculated through k-means clustering and the pseudo inverse matrix solution was used to determine the output weight matrix. The performance of the RBF classifiers can significantly be improved by an additional third optimization procedure in order to fine-tune all network parameters simultaneously. All **3-Phase-RBF** classifiers perform better as the corresponding **2-Phase-RBF** classifiers. The **3-Phase-RBF** classifiers perform as well as other regression based methods such as MLPs or polynomials. This is not surprising, as RBFs, MLPs and polynomials are approximation schemes dense in the space of continuous functions.

Table 1
Results for the hand-written digits on the test set of 10,000 examples (disjoint from the training set of 10,000 examples. Results are given as the median of three training and test runs

Classifier	Accuracy (%)
1NN	97.68
LVQ	96.99
D-Tree	91.12
2-Phase-RBF (data points)	95.24
2-Phase-RBF (k-means)	96.94
2-Phase-RBF (LVQ)	95.86
2-Phase-RBF (D-Tree)	92.72
3-Phase-RBF (data points)	97.23
3-Phase-RBF (k-means)	98.06
3-Phase-RBF (LVQ)	98.49
3-Phase-RBF (D-Tree)	94.38
SV-RBF	98.76
MLP	97.59
Poly40	98.64
3-Phase-RBF40 (LVQ)	98.45
SV-RBF40	98.56

The **1NN** and **LVQ** classifiers perform surprisingly well, particularly in comparison with RBF classifiers trained only in two phases. The **SV-RBF** and **SV-RBF40** classifiers perform very well in our numerical experiments. We found no significant difference between the classifiers on the 256-dimensional data set and the data set reduced to the 40 principal components.

The error rates for **SV-RBF**, **SV-RBF40**, **Poly40** and for RBF classifiers trained through three-phase learning with LVQ prototype initialization **3-Phase-RBF** and **3-Phase-RBF40** are very good. Although the performances of the **SV-RBF** and **3-Phase-RBF** classifiers are similar, the architectures are completely different. The complete **SV-RBF** classifier architecture consists of 10 classifiers, where approximately 4200 support vectors are selected from the training data set. In contrast, the **3-Phase-RBF** classifiers with a single hidden layer contain only 200 representative prototypes distributed over the whole input space.

An interesting property of RBF networks is that the centers in an RBF network are typical feature vectors and can be considered as representative patterns of the data set, which may be displayed and analyzed in the same way as the data.

In Figs. 4 and 5, a set of 60 RBF centers is displayed in the same style as the data points shown in Fig. 3. Here, for each digit a subset of six data points was selected at random from the training set. Each of these 10 subsets serves as seed for the cluster centers of an incremental k-means clustering

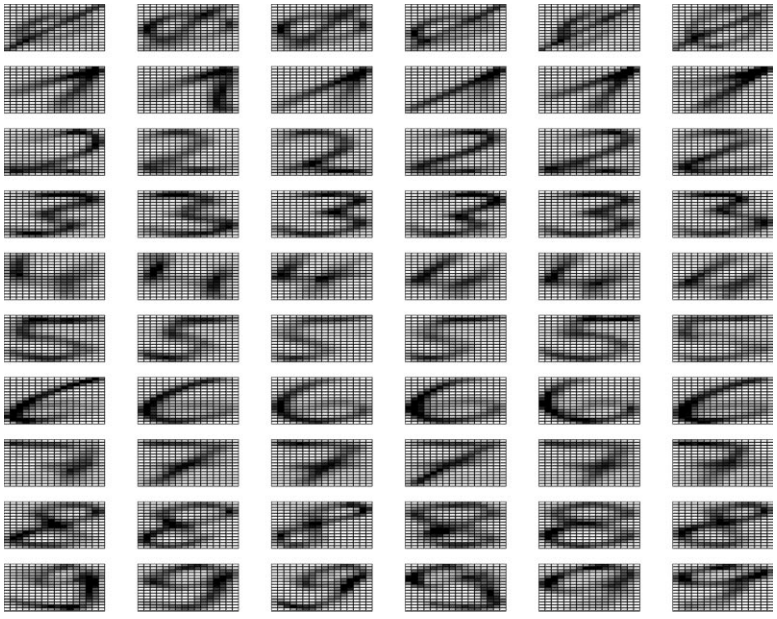

Fig. 5. The 60 RBF centers of the hand-written digits after the third backpropagation learning phase of the RBF network. Cluster centers shown in Fig. 4 are used as the initial location of the RBF centers.

procedure. After clustering the data of each digit, the union of all 60 cluster centers is used as RBF centers, the scaling parameters are calculated and the output layer weights are adapted in a second training phase as described in Section 2. These RBF centers are shown in Fig. 4.

The whole set of parameters in RBF network is then trained simultaneously by backpropagation for 100 training epochs, see Section 3. During this third training phase, the RBF centers slightly changed their initial locations. These fine-tuned RBF centers are depicted in Fig. 5. Pairs of corresponding RBF centers of Figs. 4 and 5 are very similar. The distance between these pairs of centers before and after the third learning phase was only $\|\Delta \mathbf{c}_j\| \approx 460$ in the mean, which is significantly smaller than the distances of centers representing the same class (before the third learning phase: 1116 (mean), and after the third learning phase 1153 (mean)) and, in particular, smaller than the distances of centers representing two different classes (before the third learning phase: 1550 (mean), and after the third learning phase: 1695 (mean)). But, calculating the distance matrices of these two sets of centers in order to analyze the distance relations between the RBF centers in more detail, it can be observed that the RBF centers were adapted during this third backpropagation learning phase.

The distance matrices of the centers are visualized as matrices of gray values. In Fig. 6 the distance matrices of the RBF centers before (left panel) and after the third learning phase (right panel) are shown. In the left distance matrix, many entries with small distances between prototypes of different classes can be observed, particularly between the digits 2, 3, 8 and 9, see Fig. 6. These smaller distances between prototypes of different classes typically lead to misclassifications of data points between these classes, therefore such a set of classes is called a *confusion*

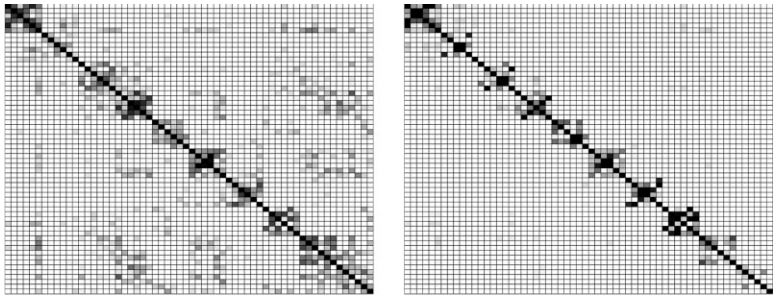

Fig. 6. Distance matrices (Euclidean distance) of 60 RBF centers before (left) and after (right) the third learning phase in an RBF network. The centers $\mathbf{c}_j$ are sorted by the classmemberships in such a way that the centers $c_1, \ldots, c_6$ are representing the digit 0, centers $c_7, \ldots, c_{12}$ are representing the digit 1, etc. Distances $d(\mathbf{c}_i, \mathbf{c}_j)$ between the centers are encoded through gray values—the smaller the distance $d(\mathbf{c}_i, \mathbf{c}_j)$ the darker is the corresponding entry in the gray value matrix. In the left distance matrix, many small distances from centers of different classes can be observed, in particular the distances between centers of the digits {2, 3, 8, 9} are very small. These small distances outside diagonal blocks often lead to misclassifications. These cannot be found in the distance matrix after three-phase learning (right figure).

Fig. 7. Examples of class bucket of the data set (left) and the calculated region of interest (right).

class. After the third learning phase of this RBF network, the centers are adjusted in such a way that these smaller distances between prototypes of different classes disappear, see Fig. 6 (right panel). This effect, dealing with the distance relations of RBF centers, cannot easily be detected on the basis of the gray value images (Figs. 4 and 5).

4.2. Application to 3D visual object recognition

The recognition of 3D objects from 2D camera images is one of the most important goals in computer vision. There is a large number of contributions to this field of research from various disciplines, e.g. artificial intelligence and autonomous mobile robots (Brooks, 1983; Lowe, 1987), artificial neural networks (Little, Poggio, & Gamble, 1988; Poggio & Edelman, 1990; Schiele & Crowley, 1996), computer vision and pattern recognition (Basri, 1996; Edelman & Duvdevani-Bar, 1997; Marr, 1982; Marr & Nishihara, 1978; Ullmann, 1996), psychophysics and brain theory (Bülthoff, Edelman, & Tarr, 1995; Edelman & Bülthoff, 1992; Logothetis & Scheinberg, 1996). Due to the increasing performance of current computer systems and the increasing development of computer vision and pattern recognition techniques, several 3D object recognition systems have been developed (Lades, Vorbrüggen, Buhmann, Lange, v.d. Malsburg, Würtz et al., 1993; Mel, 1997; Murase & Nayar, 1995; Papageorgiou & Poggio, 2000; Zhu & Yuille, 1996). The recognition of a 3D object consisted of the following three subtasks (details on this application may be found in Schwenker & Kestler, 2000).

1. **Localization of objects in the camera image**. In this processing step the entire camera image is segmented into regions, see Fig. 7. Each region should contain exactly one single 3D object. Only these marked regions, which we call the regions of interest (ROI), are used for the further image processing steps. A color-based approach for the ROI-detection is used.
2. **Extraction of characteristic features**. From each ROI within the camera image, a set of features is computed. For this, the ROIs are divided into $n \times n$ subimages and for each subimage an orientation histogram with eight orientation bins is calculated from the gray valued image. The orientation histograms of all subimages are concatenated into the characterizing feature vector, see Fig. 8, here n is set equal to 3. These feature vectors are used for classifier construction in the training phase, and are applied to the trained classifier during the recognition phase.
3. **Classification of the extracted feature vectors**. The extracted feature vectors together with the target classification are used in a supervised learning phase to build the neural network classifier. After network training novel feature vectors are presented to the classifier which outputs the estimated class labels.

4.2.1. Data

Camera images were recorded for six different 3D objects (orange juice bottle, small cylinder, large cylinder, cube, ball and bucket) with an initial resolution of 768×576 pixels. To these objects, nine different classes were assigned (bottle lying/upright), cylinders lying/upright). The test scenes were acquired under mixed natural and artificial lighting, see Fig. 9. Regions of interest were calculated

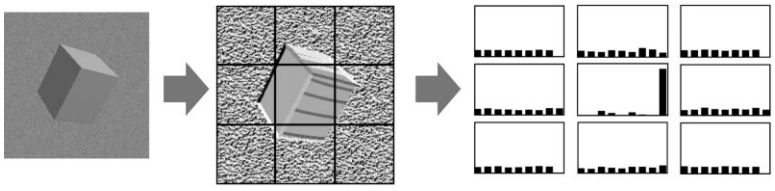

Fig. 8. Elements of the feature extraction method. From the gray valued image (left) gradient image (center; absolute value of the gradient) is calculated. Orientation histograms (right) of non-overlapping subimages constitute the feature vector.

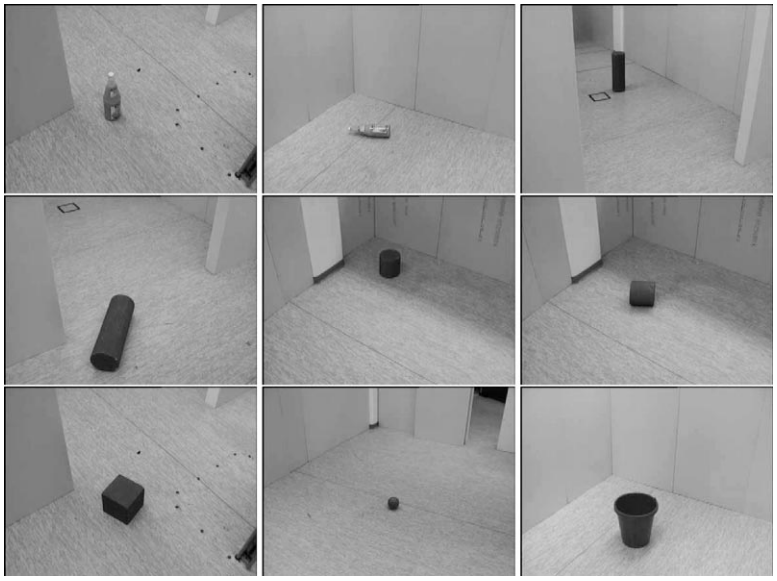

Fig. 9. Examples of the real world data set (class 0/1, orange juice bottle upright/lying; class 2/3, large cylinder upright/lying; class 4/5, small cylinder upright/lying; class 6, cube; class 7, ball; class 8, bucket).

from 1800 images using color blob detection. These regions were checked and labeled by hand, 1786 images remained for classifier evaluation. Regions of interest are detected using three color ranges, one for red (bucket, cylinder, ball), blue (cylinder) and yellow (cylinder, bucket, orange juice). The image in Fig. 7 gives an example of the automatically extracted region of interest. Features were calculated from concatenated 5×5 histograms with 3×3 Sobel operator, see Fig. 8. This data set of 1786 feature vectors of $\mathbb{R}^{200}$ serves as the evaluation data set.

4.2.2. Results

In this application for the LVQ classifiers and the RBF networks initialized through randomly selected data points, through prototypes calculated by clustering or vector quantization, 90 centers (10 per class) have been used in the numerical experiments. The decision tree classifiers have been trained by Quinlan's C4.5 algorithm. The decision trees had approximately 60 leaves in the mean, so the resulting RBF networks have approximately 60 centers, see Table 2.

As in the application to hand-written digits, the **1NN** and particularly the **LVQ** classifiers perform very well. The error rate of the **LVQ** classifier was lower than all **2-Phase-RBF** classifiers, surprisingly also better than the RBF network initialized with the LVQ prototypes and additional output layer training. As already observed in the OCR application, the performance of the **2-Phase-RBF** classifiers can significantly be improved by an additional third back-propagation-like optimization procedure. All **3-Phase-RBF** classifiers perform better as the corresponding **2-Phase-RBF** classifiers. The decision tree architectures **D-Tree**, **2-Phase-RBF (D-Tree)**, and **3-Phase-RBF (D-Tree)** show very poor classification results. This is due to the fact that the classifying regions given through the tree's leaves are determined through a few features, in the experiments approximately only eight features in the mean. In this application, the best classification results were achieved with the **SV-RBF** classifier and the **3-Phase-RBF (LVQ)** trained through three-phase learning with LVQ prototype initialization.

In Fig. 10, the distance matrices of $9 \times 6 = 54$ RBF centers before (left panel) and after (right panel) the third learning phase of the RBF network are shown. The RBF centers were calculated as for the application to hand-written digits, see Section 4.1. In both distance matrices a large confusion class can be observed, containing the classes 2–6 and 8. The centers of class 7 are separated from the centers of the other classes. After the third learning phase of the RBF network, these distances between centers of different classes become a little larger.

Table 2
Classification results of the camera images. The mean of five 5-fold cross-validation runs and the standard deviation is given

Classifier	Accuracy (%)
1NN	90.51 ± 0.17
LVQ	92.70 ± 0.71
D-Tree	78.13 ± 1.21
2-Phase-RBF (data points)	87.72 ± 0.65
2-Phase-RBF (*k*-means)	88.16 ± 0.30
2-Phase-RBF (LVQ)	92.10 ± 0.40
2-Phase-RBF (D-Tree)	77.13 ± 1.18
3-Phase-RBF (data points)	89.96 ± 0.36
3-Phase-RBF (*k*-means)	92.94 ± 0.47
3-Phase-RBF (LVQ)	93.92 ± 0.19
3-Phase-RBF (D-Tree)	77.60 ± 1.17
SV-RBF	93.81 ± 0.18

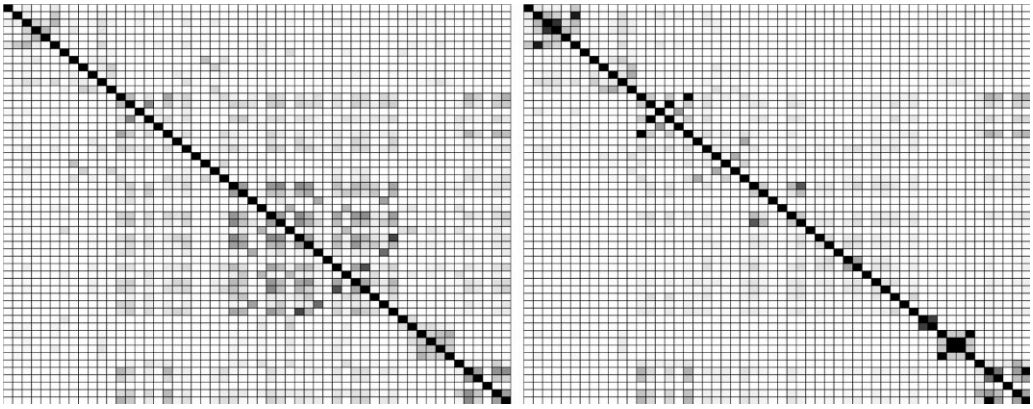

Fig. 10. The distance matrices of $9 \times 6 = 54$ RBF centers before (left panel) and after (right panel) the third learning phase in an RBF network are given. Centers $\mathbf{c}_j$ are sorted by the classmemberships in such a way that the centers $c_1, \ldots, c_6$ are representing class 0, centers $c_7, \ldots, c_{12}$ representing class 1, etc. The distances $d(\mathbf{c}_i, \mathbf{c}_j)$ are encoded through gray values. In the distance matrix calculated before the third learning phase has been started (left), many small distances for centers of different classes can be observed.

This can be observed in Fig. 10 (right panel) where the number of small distances outside the diagonal blocks is reduced.

4.3. Application to high-resolution ECGs

In this section, RBF networks are applied to the classification of high-resolution electrocardiograms (ECG). The different training schemes for RBF classifiers have been tested on data extracted from the recordings of 95 subjects separated into two groups. Two completely different types of feature extraction have been used—the ventricular late potential analysis and the beat-to-beat ECGs. Thus, we present results for two different sets of feature vectors (see Kestler & Schwenker, 2000, for further details).

4.3.1. Background

The incidence of sudden cardiac death (SCD) in the area of the Federal Republic of Germany is about 100,000 to 120,000 cases per year. Studies showed that the basis for a fast heartbeat which evolved into a heart attack is a localized damaged heart muscle with abnormal electrical conduction characteristics. These conduction defects, resulting in an abnormal contraction of the heart muscle may be monitored by voltage differences of electrodes fixed to the chest. High-resolution electrocardiography is used for the detection of fractionated micropotentials, which serve as a non-invasive marker for an arrhythmogenic substrate and for an increased risk for malignant ventricular tachyarrhythmias.

4.3.2. Ventricular late potential analysis.

Ventricular late potential (VLP) analysis is herein the generally accepted non-invasive method to identify patients with an increased risk for reentrant ventricular tachycardias and for risk stratification after myocardial infarction (Höher & Hombach, 1991). Signal-averaged high-resolution ECGs are recorded from three orthogonal bipolar leads (X, Y, Z). These are filtered and a vector magnitude V is calculated, see Fig. 11.

From this vector magnitude signal V three features are extracted:

- *QRSD* (QRS duration):
  $$QRSD := QRS_{\text{offset}} - QRS_{\text{onset}}$$
- *RMS* (Time A: $= QRS_{\text{offset}} - 40$ ms):
  $$RMS := \sqrt{\frac{1}{QRS_{\text{offset}} - A} \sum_{i=A}^{QRS_{\text{offset}}} V_i^2}$$
- *LAS* (Duration of the low amplitude signal below 40 μV):
  $$LAS := QRS_{\text{offset}} - \operatorname{argmax}\{i | V_i \geq 40\ \mu V\}$$

The three features are used as inputs to the classifiers, which are trained to predict the group status. In standard late potential analysis, a subject is termed *VLP positive* if two of the following three criteria are met: $QRSD > 115$ ms, $RMS < 20$ μV, $LAS > 38$ ms, (Breithardt, Cain, El-Sherif, Flowers, Hombach, Janse et al., 1991; Höher & Hombach, 1991).

4.3.3. Beat-to-beat ECG recordings

High-resolution beat-to-beat ECGs of 30 min duration were recorded during sinus rhythm from bipolar orthogonal X, Y, Z leads using the same equipment as with the signal-averaged recordings. Sampling rate was reduced to 1000 Hz. QRS triggering, reviewing of the ECG, and arrhythmia detection were done on a high-resolution ECG analysis platform developed by our group (Ritscher, Ernst, Kammrath, Hombach, & Höher, 1997). The three leads were summed into a signal $V = X + Y + Z$. From each recording, 250 consecutive sinus beats preceded by

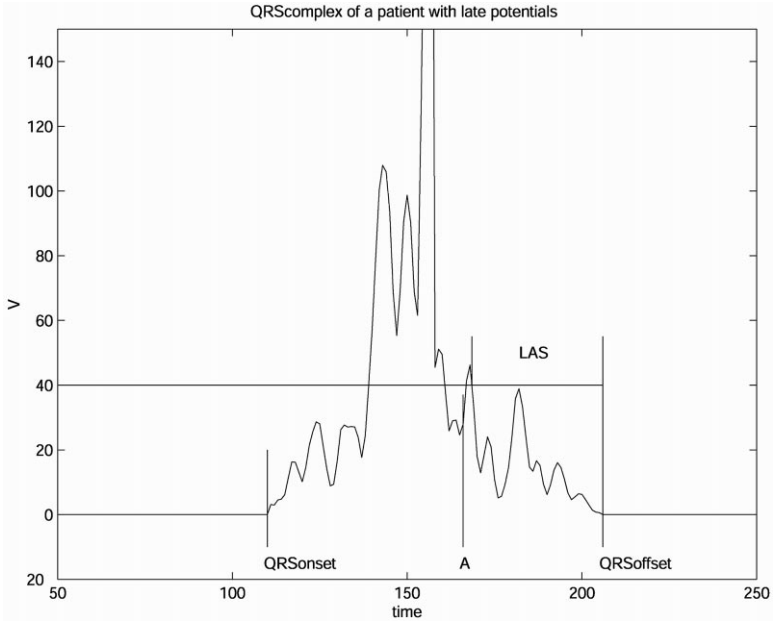

Fig. 11. Signal-averaged ECG: example of the vector magnitude signal V of a patient with late potentials.

another sinus beat were selected for subsequent beat-to-beat variability analysis.

In a first step, the signals were aligned by maximizing the cross-correlation function (van Bemmel & Musen, 1997) between the first and all following beats. Prior to the quantification of signal variability, the beats were pre-processed to suppress the main ECG waveform, bringing the beat-to-beat micro-variations into clearer focus. To achieve this, the individual signal was subtracted from its cubic spline smoothed version (spline filtering, spline interpolation through every seventh sample using the not-a-knot end condition) (de Boor, 1978; Kestler, Höher, Palm, Kochs, & Hombach, 1996). This method resembles a waveform adaptive, high-pass filtering without inducing phase-shift related artifacts. Next, for each individual beat the amplitude of the difference signal was normalized to zero mean and a standard deviation of 1 μV. Beat-to-beat variation of each point was measured as the standard deviation of the amplitude of corresponding points across all 250 beats. For the QRS we used a constant analysis window of 141 ms which covered all QRS complexes of this series (Kestler, Wöhrle, & Höher, 2000). This 141 dimensional variability vector is used as input for the classifiers.

4.3.4. Patients

High-resolution beat-to-beat recordings were obtained from 95 subjects separated into two groups. Group A consisted of 51 healthy volunteers without any medication. In order to qualify as healthy, several risk factors and illnesses had to be excluded. Group B consisted of 44 patients. Inclusion criteria were an inducible clinical ventricular tachycardia (>30 s) at electrophysiologic study, with a history of myocardial infarction and coronary artery disease in angiogram, see Kestler et al. (2000).

4.3.5. Results

For the LVQ classifiers, 12 prototypes (six per class) are used. The RBF networks initialized through randomly selected data points, or through prototypes calculated by clustering or vector quantization methods also consisted of 12 RBF centers. The decision tree classifiers trained by Quinlan's C4.5 algorithm lead to RBF networks with approximately two RBF centers (in the mean) for the three input features and to approximately eight RBF centers (in the mean) for the 141 dimensional times series.

Several topics were touched on in this investigation: the role of non-invasive risk assessment in cardiology; new signal processing techniques utilizing not only the three standard VLP parameters, but also, processing sequences of beats; and the application of RBF networks in this assessment.

By using the more elaborate categorization methods of RBF networks compared to VLP assessment on the three dimensional signal-averaged data, an increase in accuracy of about 10% could be gained (VLP results: Acc = 72.6%, Sensi = 63.6%, Speci = 80.4%) for all classifiers (Table 3). Accuracy, sensitivity, and specificity are used for the classifier evaluation (Fig. 12). For this data set the classification rate of the **LVQ** and all **RBF** classifiers are more or less the same. In comparison with the other applications, the decision tree architectures **D-Tree**, **2-Phase-RBF (D-Tree)**, and **3-Phase-RBF (D-Tree)** show good classification results. Unfortunately, the sensitivity of all methods on the 3D data is still too low to qualify as a single screening test.

In the case of the 141 dimensional beat-to-beat variability

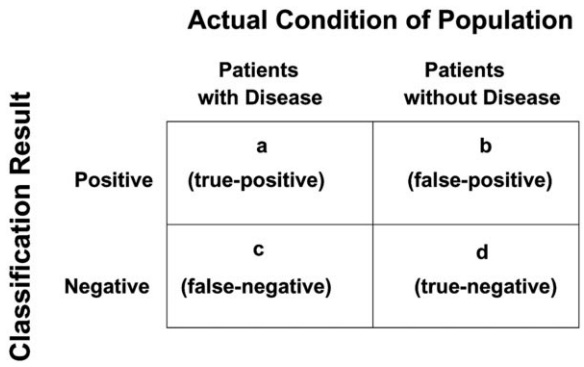

Fig. 12. Confusion matrix of a two class pattern recognition problem. The classification results for the ECG applications are typically given in terms of the following measures of performance, accuracy (acc), sensitivity (sensi), specificity (speci), positive predictive value (PPV), and the negative predictive value (NPV). These are defined through: $acc = \frac{a+d}{a+b+c+d}$, $sensi = \frac{a}{a+c}$, $speci = \frac{d}{b+d}$, $PPV = \frac{a}{a+b}$, and $NPV = \frac{d}{c+d}$. We use the accuracy, sensitivity, and specificity for the classifier evaluation.

data, there is also a substantial increase (7–15%) in classification accuracy (see Table 4) compared with categorization via a single cut-off value on the sum of the variability features (10-fold cross-validation) (mean ± stdev): Acc = 68.9 ± 5%, Sensi = 66.1 ± 8.7%, Speci = 71.4 ± 16.8%) (Kestler et al., 2000).

For the beat-to-beat data set, the **1NN** and the **LVQ** classifiers perform very well. As for the OCR and the object recognition application, the performance of the **LVQ** classifiers was better than all **2-Phase-RBF** classifiers; furthermore, the performance of the **2-Phase-RBF** classifiers can be significantly improved by an additional third learning phase. All **3-Phase-RBF** classifiers perform better as the corresponding **2-Phase-RBF** classifiers. In this application the best classification results were achieved with the **SV-RBF** classifier and the **3-Phase-RBF** trained through three-phase learning with LVQ or *k*-means center initialization.

In Figs. 13 and 14 the distance matrices are shown for the 3-dimensional signal-averaged data and 141 dimensional beat-to-beat variability data. For both data sets, $2 \times 6 = 12$ RBF centers were used. The distance matrices of the RBF centers were calculated as described in Section 4.1 and are shown before (left panels) and after the third learning phase (right panels) of the RBF network. For the 3-dimensional signal-averaged data, only a small difference between the two distance matrices can be observed. But, for the 141 dimensional beat-to-beat variability data the third backpropagation learning phase leads to a significant reduction of the small distances between RBF centers of different classes.

5. Conclusion

In this paper, algorithms for the training of RBF networks have been presented and applied to build RBF classifiers for three completely different real world applications in pattern recognition: (a) the classification of visual objects (3D objects); (b) the recognition of hand-written digits (2D objects); and (c) the classification of high-resolution electrocardiograms (1D objects).

We have discussed three different types of RBF learning schemes: two-phase, three-phase, and support vector learning. For two- and three-phase learning, three different algorithms for the initialization of the first layer of an RBF network have been presented: *k*-means clustering, learning vector quantization, and classification trees. This first step of RBF learning is closely related to density estimation, in particular when unsupervised clustering methods are used. In learning phases two and three, an error criterion measuring the difference between the network's output and the target output is minimized. In the context of learning in RBF networks, we considered support vector learning as a special type of one-phase learning scheme.

Using only the first two phases is very common, see the studies on machine learning algorithms (Michie et al., 1994; Lim, Loh, & Shih, 2000). This has led to the prejudice that MLPs often outperform RBF networks. Our experience in the use of RBF networks for these pattern recognition tasks shows that in most cases the performance of the RBF network can be improved by applying the gradient descent after an initial application of the first two learning phases. Therefore, the most economical approach simply uses the

Table 3
Classification results for the VLP input features (three features). The mean of ten 10-fold cross-validation simulations and the standard deviation is given

Classifier	Accuracy (%)	Sensitivity (%)	Specificity (%)
1NN	76.10 ± 2.21	79.93 ± 2.14	72.82 ± 3.52
LVQ	87.36 ± 2.44	75.30 ± 2.02	97.49 ± 1.92
D-Tree	87.26 ± 0.99	75.30 ± 1.76	95.44 ± 1.66
2-Phase-RBF (data points)	87.36 ± 2.44	77.39 ± 1.64	96.04 ± 2.20
2-Phase-RBF (*k*-means)	86.94 ± 2.16	76.54 ± 3.12	95.88 ± 2.26
2-Phase-RBF (LVQ)	87.36 ± 1.56	75.99 ± 2.52	96.88 ± 1.33
2-Phase-RBF (D-Tree)	88.10 ± 1.05	79.35 ± 1.69	95.65 ± 1.25
3-Phase-RBF (data points)	87.15 ± 1.03	77.38 ± 1.83	95.63 ± 2.00
3-Phase-RBF (*k*-means)	86.63 ± 2.35	76.54 ± 3.74	95.30 ± 1.56
3-Phase-RBF (LVQ)	87.36 ± 1.15	75.99 ± 2.05	96.89 ± 1.29
3-Phase-RBF (D-Tree)	88.10 ± 1.05	79.35 ± 1.69	95.65 ± 1.25
SV-RBF	88.36 ± 0.70	77.39 ± 1.69	96.04 ± 1.21

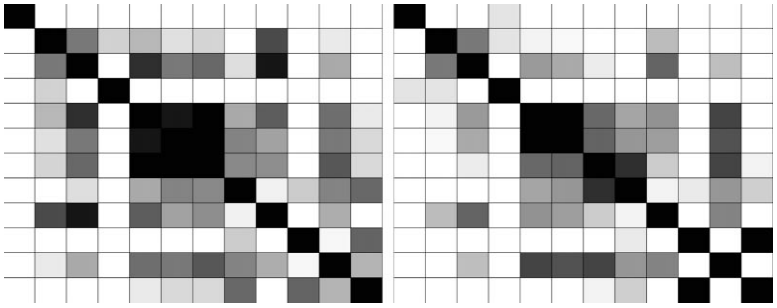

Fig. 13. Distance matrices (Euclidean distance) of 12 RBF centers for the VLP data set (three input features) before (left figure) and after (right figure) the third learning phase in an RBF network. The centers $\mathbf{c}_j$ are sorted by the classmemberships in such a way that the centers $c_1, \ldots, c_6$ are representing class 0 and centers $c_7, \ldots, c_{12}$ representing class 1. Distances $d(\mathbf{c}_i, \mathbf{c}_j)$ are encoded through gray values.

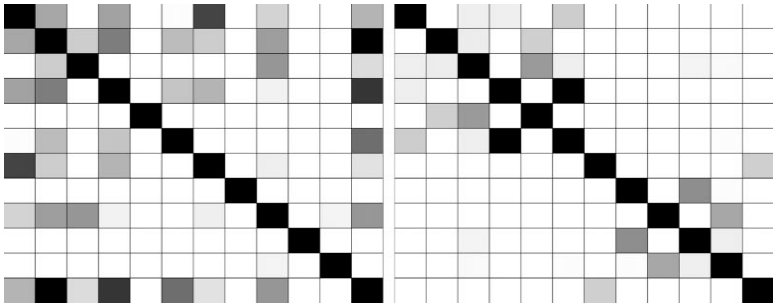

Fig. 14. Distance matrices (Euclidean distance) of 12 RBF centers for the beat-to-beat data set (141 dimensional input features) before (left panel) and after (right panel) the learning phase in an RBF network. The centers $\mathbf{c}_j$ are sorted by the classmemberships in such a way that the centers $c_1, \ldots, c_6$ are representing class 0 and centers $c_7, \ldots, c_{12}$ representing class 1. Distances $d(\mathbf{c}_i, \mathbf{c}_j)$ are encoded through gray values. After the third training phase small distances between centers of different class cannot be observed.

first two steps as a fast way of computing a good initialization for the final gradient descent. In our applications, the performance of classifiers trained by two-phase learning was improved through a third learning phase. The performance of RBF networks trained by three-phase learning and support vector learning is comparable, but RBF networks trained by support vector learning often lead to a more complex network structure. A practical advantage of two- and three-phase learning in RBF networks is the possibility to use unlabeled training data and clustering algorithms for the first training phase of the RBF centers.

Comparing RBF networks with MLP networks with one hidden layer, there seems to be a lot of similarity at first sight. Both types of networks are based on a universal approximation scheme, where the network complexity simply increases with the number of hidden MLP- or RBF units. In both cases, we have the statistical problem of choosing the right model complexity, also called the bias–variance dilemma (Geman, Bienenstock & Doursat, 1993), but there are important differences. Supervised optimization, usually implemented as back-propagation or one of its variants, is essentially the only resource for training an

Table 4
Classification results for the beat-to-beat input features (141 features). The mean of ten 10-fold cross-validation simulations and the standard deviation is given

Classifier	Accuracy (%)	Sensitivity (%)	Specificity (%)
1NN	77.57 ± 2.29	62.44 ± 3.14	87.00 ± 2.67
LVQ	76.10 ± 2.44	61.55 ± 2.87	88.92 ± 3.46
D-Tree	71.68 ± 1.72	70.74 ± 4.95	72.51 ± 5.15
2-Phase-RBF (data points)	74.73 ± 2.44	67.39 ± 3.01	80.92 ± 2.55
2-Phase-RBF (*k*-means)	75.68 ± 3.71	68.40 ± 4.92	81.97 ± 3.80
2-Phase-RBF (LVQ)	74.94 ± 2.52	69.42 ± 5.34	79.81 ± 3.33
2-Phase-RBF (D-Tree)	73.89 ± 3.25	72.31 ± 4.69	75.19 ± 4.27
3-Phase-RBF (data points)	76.10 ± 2.58	69.90 ± 2.91	81.28 ± 3.02
3-Phase-RBF (*k*-means)	77.05 ± 1.87	71.79 ± 2.29	81.62 ± 2.92
3-Phase-RBF (LVQ)	77.05 ± 2.52	71.41 ± 3.56	82.00 ± 3.86
3-Phase-RBF (D-Tree)	74.52 ± 3.04	70.32 ± 4.84	78.17 ± 3.31
SV-RBF	79.05 ± 1.26	71.79 ± 2.36	82.38 ± 2.87

MLP network. There is no option for training the two network layers separately and there is no opportunity of network initialization as in RBF networks. MLP units in the hidden layer can be viewed as soft decision hyperplanes defining certain composite features that are then used to separate the data as in a decision tree. The RBF units, on the other hand, can be viewed as smoothed typical data points.

Given a new data point, the RBF network essentially makes a decision based on the similarity to known data points, whereas the MLP network makes a decision based on other decisions. By this characterization, one is reminded of the distinction made in artificial intelligence between rule-based and case-based reasoning. It seems that the decision made by an MLP is more rule-based, whereas that made by RBF networks is more case-based. This idea is plausible, as RBF centers can indeed be recognized as representative data points and they can actually be displayed and interpreted in the same way as data points. In our applications, we observe that the RBF centers moved only a little during the gradient descent procedure, so that the units in the RBF network can still be interpreted as representative data points. This is an important property of RBF networks in applications where the classifier system has to be built by a non-specialist in the field of classifier design.

Appendix A. Support vector learning in RBF networks

Here, we give a short review on support vector (SV) learning in RBF networks (Cristianini & Shawe-Taylor, 2000; Schölkopf, Burges, & Smola, 1998; Vapnik, 1998). The support vector machine (SVM) was initially developed to classify data points of a linear separable data set. In this case, a training set consisting of M examples $(\mathbf{x}^\mu, y^\mu), \mathbf{x}^\mu \in \mathbb{R}^d$, and $y^\mu \in \{-1, 1\}$ can be divided into two sets by a separating hyperplane. Such a hyperplane is determined by a weight vector $\mathbf{w} \in \mathbb{R}^d$ and a bias or threshold $\theta \in \mathbb{R}$ satisfying the separating constraints

$$y^\mu[\langle \mathbf{x}^\mu, \mathbf{w}\rangle + \theta] \geq 1, \qquad \mu = 1, \ldots, M. \tag{44}$$

The distance between the separating hyperplane and the closest data points of the training set is called the *margin*. Intuitively, the larger the margin, the higher the generalization ability of the separating hyperplane. The optimal separating hyperplane with maximal margin is unique and can be expressed by a linear combination of those training examples lying exactly at the margin. These data points are called the *support vectors*. This separating hyperplane with maximal margin has the form

$$H(\mathbf{x}) = \sum_{\mu=1}^{M} \alpha_\mu^* y^\mu \langle \mathbf{x}, \mathbf{x}^\mu\rangle + \alpha_0^* \tag{45}$$

where $\alpha_1^*, \ldots, \alpha_M^*$ is the solution optimizing the functional

$$Q(\alpha) = \sum_{\mu=1}^{M} \alpha_\mu - \frac{1}{2} \sum_{\mu,\nu=1}^{M} \alpha_\mu \alpha_\nu y^\mu y^\nu \langle \mathbf{x}^\mu, \mathbf{x}^\nu\rangle \tag{46}$$

subject to the constraints $\alpha_\mu \geq 0$ for all $\mu = 1, \ldots, M$ and

$$\sum_{\mu=1}^{M} \alpha_\mu y^\mu = 0. \tag{47}$$

A vector of the training set $\mathbf{x}^\mu$ is a support vector if the corresponding coefficient $\alpha_\mu^* > 0$. Then, the weight vector $\mathbf{w}$ has the form

$$\mathbf{w} = \sum_{\mu=1}^{M} \alpha_\mu y^\mu \mathbf{x}^\mu = \sum_{\mu,\alpha_\mu>0} \alpha_\mu y^\mu \mathbf{x}^\mu$$

and the bias α_0^* is determined by a single support vector $(\mathbf{x}^s, y^s)$:

$$\alpha_0^* = y^s - \langle \mathbf{w}, \mathbf{x}^s\rangle.$$

The SVM approach has been extended to the non-separable situation and to the regression problem. In most applications (regression or pattern recognition problems) linear hyperplanes as solutions are insufficient. For example, in real world pattern recognition problems it is common to define an appropriate set of nonlinear mappings $\mathrm{g} = (g_1, g_2, \ldots)$, where the g_j are defined as real valued functions, transforming an input vector $\mathbf{x}^\mu$ to a vector $g(\mathbf{x}^\mu)$ which is an element of a new feature space $\mathcal{H}$. Then, the separating hyperplane can be constructed in this new feature space $\mathcal{H}$ and can be expressed by

$$H(\mathbf{x}) = \sum_{\mu=1}^{M} \alpha_\mu y^\mu \langle g(\mathbf{x}), g(\mathbf{x}^\mu)\rangle + \alpha_0. \tag{48}$$

Provided $\mathcal{H}$ is a Hilbert space and K a kernel $K: \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ satisfying the condition of Mercer's theorem an explicit mapping $g: \mathbb{R}^d \to \mathcal{H}$ does not need to be known, because it is implicitly given through

$$K(\mathbf{x}, \mathbf{x}^\mu) = \langle g(\mathbf{x}), g(\mathbf{x}^\mu)\rangle.$$

Kernel function K is representing the inner product between vectors in the feature space $\mathcal{H}$. With a suitable choice of a kernel function, the data can become separable in feature space despite being not separable in the input space. Using such a kernel function K, the separating hyperplane is given by

$$H(\mathbf{x}) = \sum_{\mu=1}^{M} \alpha_\mu y^\mu K(\mathbf{x}, \mathbf{x}^\mu) + \alpha_0.$$

The coefficients α_μ can be found by solving the optimization problem

$$Q(\alpha) = \sum_{\mu=1}^{M} \alpha_\mu - \frac{1}{2} \sum_{\mu,\nu=1}^{M} \alpha_\mu \alpha_\nu y^\mu y^\mu K(\mathbf{x}^\nu, \mathbf{x}^\nu)$$

subject to the constraints $0 \leq \alpha_\mu \leq C$ for all $\mu = 1, \ldots, M$

and

$$\sum_{\mu=1}^{M} \alpha_\mu y^\mu = 0$$

where C is a predefined positive number. An important kernel function satisfying Mercer's condition is the Gaussian kernel function

$$K(\mathbf{x}, \mathbf{y}) = \mathrm{e}^{-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2}.$$

The separating surface obtained by the SVM approach is a linear combination of Gaussian functions located at the support vectors. The SVM reduces to an RBF network. In contrast to RBF networks described previously, the centers are now located at certain data points of the training set and the number of centers is automatically determined in this approach. Furthermore, all Gaussians are radially symmetric, all with the same kernel width σ^2.

References

Armijo, L. (1966). Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, *16* (1), 1–3.

Basri, R. (1996). Recognition by prototypes. *International Journal of Computer Vision*, *19*, 147–168.

Bishop, C. M. (1995). *Neural networks for pattern recognition*, Oxford: Clarendon Press.

Breithardt, G., Cain, M., El-Sherif, N., Flowers, N., Hombach, V., Janse, M., Simson, M., & Steinbeck, G. (1991). Standards for analysis of ventricular late potentials using high resolution or signal-averaged electrocardiography. *Eur. Heart J.*, *12*, 473–480.

Brooks, R. (1983). Model-based three-dimensional interpretations of two-dimensional images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *5*, 140–149.

Broomhead, D., & Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, *2*, 321–355.

Bülthoff, H., Edelman, S., & Tarr, M. (1995). How are three-dimensional objects represented in the brain? *Cerebral Cortex*, *5*, 247–260.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines*, Cambridge, UK: Cambridge University Press.

Darken, C. & Moody, J. (1990). Fast adaptive *k*-means clustering: some empirical results. *Proceedings International Joint Conference on Neural Networks*. Piscataway: IEEE Press, 238.

de Boor, C. (1978). *A practical guide to splines*, New York: Springer Verlag.

Edelman, S., & Bülthoff, H. (1992). Orientation dependence in the recognition of familiar and novel views of three-dimensional objects. *Vision Research*, *32*, 2385–2400.

Edelman, S., & Duvdevani-Bar, S. (1997). A model of visual recognition and categorization. *Phil. Trans. R. Soc. London B*, *352*, 1191–1202.

Geman, S., Bienenstock, E., & Doursat, R. (1993). Neural networks and the bias/variance dilemma. *Neural Computation*, *4*, 1–58.

Ghitza, O. (1991). Auditory nerve representation as a basis for speech recognition. In S. Furui & M. Sondhi, *Advances in speech signal processing* (pp. 453–485). New York: Marcel Dekker.

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

Hertz, J., Krogh, A., & Palmer, R. G. (1991). *Introduction to the theory of neural computation*, New York: Addison Wesley.

Höher, M., & Hombach, V. (1991). Ventrikuläre Spätpotentiale—Teil II Klinische Aspekte. *Herz & Rhythmus*, *3* (4), 8–14.

Kestler, H., & Schwenker, F. (2000). Classification of high-resolution ECG signals. In R. Howlett & L. Jain, *Radial basis function neural networks: theory and applications*, Heidelberg: Physica-Verlag (in press).

Kestler, H. A., Höher, M., Palm, G., Kochs, M., & Hombach, V. (1996). Time domain variability of high resolution beat-to-beat recordings classified by neural networks. In A. Murray & R. Arzbaecher, *Computers in cardiology* (pp. 317–320). Piscataway: IEEE Computer Society.

Kestler, H. A., Wöhrle, J., & Höher, M. (2000). Cardiac vulnerability assessment from electrical microvariability of the high-resolution electrocardiogram. *Medical & Biological Engineering & Computing*, *38*, 88–92.

Kohonen, T. (1990). The self-organizing map. *Proc. IEEE*, *78* (9), 1464–1480.

Kohonen, T. (1995). *Self-organizing maps*, Berlin: Springer.

Kubat, M. (1998). Decision trees can initialize radial-basis-function networks. *IEEE Transactions on Neural Networks*, *9*, 813–821.

Lades, M., Vorbrüggen, J., Buhmann, J., Lange, J., v.d. Malsburg, C., Würtz, R., & Konen, W. (1993). Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, *42*, 300–311.

Light, W. (1992). Some aspects of radial basis function approximation. In S. Singh, *Approximation theory, spline functions and applications* (pp. 163–190). Dordrecht: Kluwer.

Lim, T. -J., Loh, W. -Y., & Shih, Y. -S. (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, *40*, 203–228.

Little, J., Poggio, T., & Gamble, E. (1988). Seeing in parallel: the vision machine. *International Journal of Supercomputing Applications*, *2*, 13–28.

Logothetis, N., & Scheinberg, D. (1996). Visual object recognition. *Annual Review of Neuroscience*, *19*, 577–621.

Lowe, D. (1987). Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, *31*, 355–395.

Magoulas, G., Vrahatis, M., & Androulakis, G. (1997). Effective backpropagation training with variable stepsize. *Neural Networks*, *10* (1), 69–82.

Marr, D. (1982). *Vision*, San Francisco: Freeman.

Marr, D., & Nishihara, H. (1978). Representation and recognition of the spatial organization of three dimensional structure. *Proceedings of the Royal Society of London B*, *200*, 269–294.

Mel, B. (1997). Seemore: combining colour, shape, and texture histogramming in a neurally-inspired approach to visual object recognition. *Neural Computation*, *9*, 777–804.

Micchelli, C. (1986). Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive Approximation*, *2*, 11–22.

Michie, D., Spiegelhalter, D., & Taylor, C. (1994). *Machine learning, neural and statistical classification*, New York: Ellis Horwood.

Moody, J., & Darken, C. J. (1989). Fast learning in networks of locally-tunes processing units. *Neural Computation*, *1*, 284–294.

Murase, H., & Nayar, S. (1995). Visual learning and recognition of 3d objects from appearance. *International Journal of Computer Vision*, *14*, 5–24.

Papageorgiou, C., & Poggio, T. (2000). A trainable system for object detection. *International Journal of Computer Vision*, *38*, 15–33.

Park, J., & Sandberg, I. W. (1993). Approximation and radial basis function networks. *Neural Computation*, *5*, 305–316.

Poggio, T., & Edelman, S. (1990). A network that learns to recognize three-dimensional objects. *Nature*, *343*, 263–266.

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

Poggio, T., & Girosi, F. (1990b). Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, *2247*, 978–982.

Powell, M. J. D. (1992). The theory of radial basis function approximation in 1990. In W. Light, *Advances in numerical analysis* (pp. 105–210), vol. II. Oxford: Clarendon Press.

Quinlan, J. (1992). *C4.5 programs for machine learning*, San Francisco: Morgan Kaufmann.

Rabiner, L., & Juant, B. -H. (1993). *Fundamentals of speech recognition*, Englewood Cliffs, NJ: Prentice Hall.

Ripley, B. (1996). *Pattern recognition and neural networks*, Cambridge, UK: Cambridge University Press.

Ritscher, D. E., Ernst, E., Kammrath, H. G., Hombach, V., & Höher, M. (1997). High-resolution E.C.G analysis platform with enhanced resolution. *Computers in Cardiology*, *24*, 291–294.

Schiele, B. & Crowley, J. (1996). Probabilistic object recognition using multidimensional receptive field histograms. In *Proceedings of the 13th International Conference on Pattern Recognition* (pp. 50–54). Piscataway: IEEE Computer Press.

Schölkopf, A., Burges, C., & Smola, A. (1998). *Advances in kernel methods—support vector learning*, Cambridge, Mass: MIT Press.

Schwenker, F. (2000). Hierarchical support vector machines for multi-class pattern recognition. In R. Howlett, & L. Jain (Eds.), *Knowledge-based intelligent engineering systems and applied technologies KES 2000*, Piscataway: IEEE Press (pp. 561–565).

Schwenker, F., & Dietrich, C. (2000). Initialization of radial basis function networks by means of classification trees. *Neural Network World*, *10*, 473–482.

Schwenker, F., & Kestler, H. (2000). 3-D visual object classification with hierarchical RBF networks. In R. Howlett & L. Jain, *Radial basis function neural networks: theory and applications*, Heidelberg: Physica-Verlag (in press).

Schwenker, F., Kestler, H., Palm, G., & Höher, M. (1994). Similarities of LVQ and RBF learning. In *Proc. IEEE International Conference SMC* (pp. 646–651).

Ullmam, S. (1996). *High-level vision. Object recognition and visual cognition*, Cambridge, MA: MIT Press.

van Bemmel, J. & Musen, M. (1997). Handbook of medical informatics Heidelberg/New York: Springer Verlag.

Vapnik, V. (1998). *Statistical learning theory*, New York: John Wiley & Sons.

Wasserman, P. (1993). *Advanced methods in neural computing*, New York: Van Nostrand Reinhold.

Zhu, S., & Yuille, A. (1996). Forms: a flexible object recognition and modeling system. *International Journal of Computer Vision*, *20*, 1–39.