Radial-Basis-Function Networks: Learning and Applications

Friedhelm Schwenker  Hans A. Kestler  Günther Palm
University of Ulm, D-89069 Ulm, Germany
Email: schwenker, kestler, palm@informatik.uni-ulm.de

Abstract

We present different training algorithms for radial basis function (RBF) networks and the behaviour of RBF classifiers in three different pattern recognition applications is presented: the classification of 3-D visual objects, high-resolution electrocardiograms and handwritten digits.

1 Introduction

RBF networks were introduced into the neural network literature by Broomhead and Lowe in 1988 [1]. 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 or cells in the visual cortex sensitive to bars oriented in a certain direction. 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. We consider multivariate functions \( f : \mathbb{R}^d \rightarrow \mathbb{R}^m \). Without loss of generality we may assume that \( m \) is equal to 1. The goal of interpolating a set of tuples \( (x^\mu, y^\mu)_{\mu = 1}^M \) is to find a function \( F : \mathbb{R}^d \rightarrow \mathbb{R} \) with \( F(x^\mu) = y^\mu \) for all \( \mu = 1, \ldots, M \), where \( F \) is an element of a predefined set (often a vector space) of functions. In the RBF approach the interpolating function \( F \) is a linear combination of basis functions:

\[
F(x) = \sum_{\mu=1}^M b_\mu h(||x - x^\mu||) + p(x) \tag{1}
\]

where \( || \cdot || \) denotes the Euclidean norm, \( b_1, \ldots, b_M \) are real numbers, \( h \) a real valued function, and \( p \) a polynomial \( p \in \Pi^d_n \) (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 \( b_1, \ldots, b_M \) and the polynomial term \( p := \sum_{i=1}^D a_ip_i \) where \( p_1, \ldots, p_D \) is a basis of \( \Pi^d_n \) and \( a_1, \ldots, a_D \) are real numbers. The function \( F \) has to satisfy the conditions:

\[
F(x^\mu) = y^\mu, \quad \mu = 1, \ldots, M
\]

and

\[
\sum_{\mu=1}^M b_\mu y^\mu = 0, \quad j = 1, \ldots, D.
\]

Sufficient conditions for the unique solvability of the interpolation problem were given by several authors e.g. see results due to Micchelli, Powell, or Light [2, 3, 4]. The function \( h \) is called a radial basis function if the interpolation problem has a unique solution. In some cases the polynomial term in equation (1) can be omitted, then the interpolation problem is equivalent to the matrix equation

\[
Hb = y \tag{2}
\]

where \( b = (b_1, \ldots, b_M) \), \( y = (y', \ldots, y'^M) \), and \( M \times M \) matrix \( H \) defined by

\[
H = (h(||x' - x'^\mu||))_{\mu, \nu = 1, \ldots, M}.
\]

Provided the inverse of \( H \) exists, the solution of the interpolation problem has the form:

\[
b = H^{-1}y. \tag{3}
\]

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

\[
h(r) = e^{-r^2/\sigma^2},
\]
\[
h(r) = (r^2 + \sigma^2)^{-1/2},
\]
\[
h(r) = (r^2 + \sigma^2)^{-1/2}.
\]

Here, \( \sigma \) 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 function \( h(||x - c||) = \exp(-||x - c||^2/\sigma^2) \) with peak at center \( c \in \mathbb{R}^d \) and decreasing as the distance from the center increases.
The solution of the exact interpolating RBF
mapping passes through every data point
\((x^*, y^*)\). In the presence of noise the exact solution of the interpolation problem is typically a function badly 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 \(H\) becomes intractable in practice. In applications, where we have to deal with many thousands of noisy data points an approximate solution to the data is more desirable than an interpolative one. Broomhead and Lowe [1] first proposed to reduce the number of basis functions in order to reduce the computational complexity. This technique produces a solution approximating instead of interpolating the data points. Furthermore, in [1] 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 [5]. This implies that RBF-networks with adjustable prototypes can also be used for classification tasks [6].

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 \(c_j\) which are not limited to data points.
2. Instead of a global scaling parameter \(\sigma > 0\) each basis function has its own scaling parameter \(\sigma_j > 0\). In some RBF network models the so-called Mahalanobis distance instead of the Euclidean distance is used. A Mahalanobis distance in \(R^d\) is defined by a positive definite matrix \(R\) and is given through
   \[
d(x, y) = (x - y)^T R (x - y)
   \]
   where \(T\) denotes the transpose of a matrix. The Mahalanobis distance becomes the Euclidean distance if \(R\) is equal to the identity matrix \(I\). In this type of RBF network every basis function has its own matrix \(R_j\), usually defined as the inverse of the covariance matrix of the data points with respect to the center \(c_j\). Such an architecture contains \(d\) parameters for each center \(c_j\) plus \(d(d + 1)/2\) parameters for each matrix \(R_j\) which have to be estimated.

2 RBF Network Training

In this section we present three different training procedures for RBF networks: Two-stage training, Backpropagation training, and Support vector training. All three network learning strategies are reviewed in the following.

2.1 Two-stage training

In a multilayer perceptron all parameters are usually adapted simultaneously by an optimization procedure. This training procedure is supervised. Learning in an RBF network may be done in two or three stages: 1.) Calculating the parameters of the RBFs, including the centers \(c_j\) and the scaling parameters \(\sigma_j\) and 2.) calculation of the output weights. To determine the centers of the RBFs typically unsupervised training procedures from clustering or vector quantization are used [7]. If the RBF network has to perform a classification task supervised training procedures to determine the RBF centers are also applicable. We present two completely different supervised algorithms to initialize the RBF centers: Kohonen’s learning vector quantization (LVQ) [8], and Quinlan’s C4.5 decision trees [9].

Unsupervised Competitive Learning

A competitive neural network consists of a single layer of \(k\) neurons. Their synaptic weight vectors \(c_1, \ldots, c_k \in \mathbb{R}^d\) divide the input space into \(k\) disjoint regions \(R_1, \ldots, R_k \subset \mathbb{R}^n\), where each set \(R_j\) is defined by
   \[
   R_j = \{ x \in \mathbb{R}^n | \| x - c_j \| = \min_{i=1, \ldots, k} \| x - c_i \| \}. \tag{4}
   \]

Such a partition of the input space is called a Voronoi tesselation where each weight vector \(c_j\) represents a representative center or prototype vector for region \(R_j\).

Presenting an input vector \(x \in \mathbb{R}^n\) to the network all neurons \(j = 1, \ldots, k\) determine their Euclidean distance to \(x\):
   \[
d_j = \| x - c_j \|. \tag{5}
   \]

Competition between the neurons is realized by searching for the minimum distance:
   \[
d_j^* = \min_{j=1, \ldots, k} d_j. \tag{6}
   \]

One of the most popular methods in cluster analysis is the \(k\)-means clustering algorithm. It is easy to prove that the error function \(E\), defined by
   \[
   E(c_1, \ldots, c_k) = \sum_{j=1}^{k} \sum_{x^* \in C_j} \| x^* - c_j \|^2, \tag{7}
   \]

34
is minimal, if each prototype $c_j$ is equal to the corresponding center of gravity of data points $C_j := R_j \cap \{x^1, \ldots, x^n\}$. In the batch mode $k$-means algorithm the prototypes are set to:

$$c_j = \frac{1}{|C_j|} \sum_{x^i \in C_j} x^i. \quad (6)$$

Online optimization of $E$ can also be realized by unsupervised competitive learning [10]:

$$\Delta c_j^t = \eta_t (x^t - c_j^t) \quad (7)$$

where $c_{j^*}$ is the closest prototype to the input $x^t$. The learning rate $\eta_t$ has to be a sequence of positive real numbers such that $\eta_t \to 0$ as $t \to \infty$, $\sum_{t=1}^{\infty} \eta_t = \infty$ and $\sum_{t=1}^{\infty} \eta_t^2 < \infty$.

Each prototype $c_1, \ldots, c_k$ of this competitive neural network serves as a center of a basis function in the RBF network.

LVQ Learning

Now we assume that a classification or pattern recognition task has to be performed by the RBF network. A training set of feature vectors $x^t$ is given where each vector is labeled with the target classification $y^t$. In this case supervised learning may be used to determine the set of prototype vectors $c_1, \ldots, c_k$.

The LVQ learning algorithm has been suggested by Kohonen [11] for vector quantization and classification tasks. From the basic LVQ 1 version, the OLVQ1, LVQ2 and LVQ3 training procedures have been derived. Presenting a vector $x^t \in \mathbb{R}^d$ together with its class membership the winning prototype $j^*$ is adapted according to the LVQ1-learning rule

$$\Delta c_j = \eta_t (y^t - c_j^*) (y^t_{j^*} - \frac{1}{2} z^t_{j^*}). \quad (8)$$

here $z^t$ is the binary output vector of the network and $y^t$ is a binary target vector coding the class membership for feature input vector $x^t$. In both vectors $z^t$ and $y^t$ exactly one component is equal to 1, all others are equal to 0. Again, $\eta_t$ is a positive decreasing learning rate and each prototype $c_1, \ldots, c_k$ serves as an RBF center [8].

Once the prototype vectors have been calculated the scaling parameters $c_j$ have to be determined, typically by a nearest neighbour heuristic. Usually $c_j$ is set to be the mean or median of the distances of the $i \leq k$ nearest prototypes.

Decision Trees

Decision trees divide the feature space $\mathbb{R}^d$ into pairwise disjoint regions $R_i$. The binary tree is the most popular type, here each node has two or zero children. A node in the decision tree is representing a region $R$ of $\mathbb{R}^d$. If a node has two children then the regions represented by the children nodes form a partition of $R$. Typically, these regions are hyperrectangles parallel to the axes of the feature space. In [9] Kubat presents a method to transform such a set of disjoint regions $R_j \subset \mathbb{R}^d$ into a set of centers $c_j \in \mathbb{R}^d$ and scaling parameters $\sigma_j \in \mathbb{R}^d$ in order to initialize a Gaussian RBF network. Kubat used Quinlan’s C4.5 software package to generate the decision trees [12].

![Decision tree](image)

Figure 1: Decision tree with classes at the leaves and the corresponding partition of the feature space with class label annotation of the regions.

In Figure 1 a classification tree and the corresponding regions which were defined by the tree’s leaves are shown. Each terminal node of the classification tree determines a rectangular region in the feature space $\mathbb{R}^2$. In the binary classification tree each branch is determined by a feature dimension $i \in \{1, \ldots, d\}$ and a boundary $b_i \in \mathbb{R}$. Typically a class is represented in more than one leaf of the tree. Each of these regions $R$ is defined
by a path through the tree starting at the root and terminating in a leaf. The number of leaves in the classification tree determines the number of hidden RBF neurons in the network.

The centers $\mu_j$ of the RBFs can be calculated in the following way: For each region

$$R_j = [a_{ij}, b_{ij}] \times \cdots \times [a_{dq}, b_{dq}]$$

an RBF center $\mu_j = (\mu_{i1}, \ldots, \mu_{id})$ is determined through

$$\mu_{ij} = (a_{ij} + b_{ij})/2 \quad (9)$$

for $i = 1, \ldots, d$ (see Figure 2).

The scaling parameters $\sigma_j$ determine how steeply the RBF decreases with growing distance from the center $\mu_j$, and are determined in such a way that all RBFs have the same value at the border of their region (see Figure 2). The size of the hyperrectangle $R_j$ defines the shape of a hyperellipsoid:

$$\sigma_j = \alpha ((b_{ij} - a_{ij}), \ldots, (b_{dq} - a_{dq})) \quad (10)$$

where the parameter $\alpha > 0$ specifies the overlap of the RBF centers.

In [9] Kubuut proposed a slightly different method: The RBF centers $\mu_j$ are placed in the middle of the region $R_j$ as in $C_1$, except $R_j$ touches the border of a feature dimension $i$. In this case $\mu_j$ is placed at this border and the scaling parameters $\sigma_j$ is multiplied by two.

![Figure 2: Regions and the RBF nodes.](image)

Output Weight Matrix

Provided that the centers $\mu_j$ and the scaling parameters $\sigma_j$ of the basis functions have been determined, the weights of the output layer can be calculated. We assume $k$ basis functions in the hidden layer of the RBF network. Let $(x^\mu, y^\mu)$, $\mu = 1, \ldots, M$ be the set of training examples with feature vector $x^\mu \in \mathbb{R}^d$ and target $y^\mu \in \mathbb{R}^m$, $H_{ij}$ the outcome of the $j$-th basis function for the $\mu$-th feature vector $x^\mu$, and $Y_{ij}$ the $j$-th component of the $\mu$-th target vector $y^\mu$. Given the matrices $H = (H_{ij})$ and $Y = (Y_{ij})$ the $k \times m$ output matrix $B$ is the result of minimizing the functional

$$E(B) = \| H \cdot B - Y \|^2.$$

The solution is given explicitly in the form $B = H^+ Y$ where $H^+$ denotes the pseudo inverse matrix of $H$ which is defined as

$$H^+ = \lim_{\alpha \to \infty} (HTH + \alpha I)^{-1}HT.$$

Provided that $(HTH)^{-1}$ is defined, the pseudo inverse matrix becomes simply $H^+ = (HTH)^{-1}HT$.

After this final step all parameters of the RBF network are determined.

2.2 Backpropagation Training

Backpropagation learning in an RBF-network stands for adapting the output weights between hidden and output layer, training the prototype vectors $c_j \in \mathbb{R}^d$ and the scaling parameters $\sigma_j$ simultaneously. This can be achieved by minimizing the least mean square error functional

$$E(c, \sigma, B) = \sum_{\mu=1}^M \| y^\mu - z^\mu \|^2 \quad (11)$$

for a training set $(x^\mu, y^\mu)$, $\mu = 1, \ldots, M$ by gradient descent optimization. Here $z^\mu$ is the output of the RBF network.

Determining the gradient direction leads to the learning rules for all types of parameters $h_p$, $c_j$ and $\sigma_j$. For a single example $(x^\mu, y^\mu)$ of the training set these weight update rules are:

$$\Delta b_{ip} = \eta y^\mu_p (y^\mu_p - z^\mu_p)$$

$$\Delta c_{ij} = \eta (x^\mu_i - c_{ij}) (H'r^\mu_j) \sum_p (y^\mu_p - z^\mu_p) b_{jp}.$$

The update rule for the scaling parameters depends on the RBF and the type of scaling (matrix, vector, or real valued parameter). For the Gaussian RBF with a single parameter per basis function the update rule for $\sigma_j$ becomes:

$$\Delta \sigma_j = \eta_p \frac{(r^\mu_j)^2 H'(r^\mu_j)}{\sigma_j^2} \sum_p (y^\mu_p - z^\mu_p) b_{jp}.$$

2.3 Support Vector Learning in RBF Networks

Here a short review on support vector learning in RBF networks is given (see [13, 14, 15]). 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 \((x^\mu, y^\mu)\), \( x^\mu \in \mathbb{R}^d \), and \( y^\mu \in \{-1, 1\} \) can be divided up into two sets by a separating hyperplane. Such a hyperplane is determined by a weight vector \( b \in \mathbb{R}^d \) and a bias or threshold \( \theta \in \mathbb{R} \) satisfying the separating constraints

\[
y^\mu \left( (x^\mu, b) + \theta \right) \geq 1 \quad \mu = 1, \ldots, M.
\]

The distance between the separating hyperplane and the closed 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. The separating hyperplane has the form

\[
H(x) = \sum_{\mu=1}^{M} \alpha^*_\mu y^\mu (x^\mu, z^\mu) + a_0
\]

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

\[
Q(\alpha) = \sum_{\mu=1}^{M} \alpha_{\mu} - \frac{1}{2} \sum_{\mu, \nu=1}^{M} \alpha_{\mu} \alpha_{\nu} y^\mu y^\nu (x^\mu, z^\nu)
\]

subject to the constraints \( \alpha_{\mu} \geq 0 \) for all \( \mu = 1, \ldots, M \) and \( \sum_{\mu=1}^{M} \alpha_{\mu} y^\mu = 0 \). Then a training vector \( z^\mu \) is a support vector if the corresponding coefficient \( \alpha^*_\mu \) is > 0. The bias \( a_0^* \) is determined by a single support vector \((x^s, y^s)\):

\[
a_0^* = y^s - \alpha_0^* y^s (x^s, z^s).
\]

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 solutions are insufficient. For example, in real world pattern recognition problems it is common to define an appropriate set of nonlinear mappings \( g = (g_1, g_2, \ldots) \), each \( g_j \) defined as a real valued function, transforming the input vectors \( x^\mu \) to a vector \( g(x^\mu) \) which is element of a new feature space \( \mathcal{H} \). Then the separating hyperplane can be constructed in the feature space \( \mathcal{H} \) and can be expressed by

\[
H(x) = \sum_{\mu=1}^{M} \alpha_{\mu} y^\mu (g(x^\mu), g(z^\mu)) + a_0.
\]

Provided \( \mathcal{H} \) is a Hilbert space, the explicit mapping \( g(x) \) does not need to be known since it can implicitly defined by a kernel function

\[
K(x, z^\mu) = \langle g(x), g(z^\mu) \rangle
\]

representing the inner product of the feature space. 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 a kernel function \( K \) satisfying the condition of Mercer’s theorem (see [15] for details), the separating hyperplane is given by

\[
H(x) = \sum_{\mu=1}^{M} \alpha_{\mu} y^\mu K(x^\mu, z^\mu) + a_0.
\]

The coefficients \( \alpha_{\mu} \) can be found by solving the optimization problem

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

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 Mercers condition is the Gaussian kernel function

\[
K(x, y) = e^{-\frac{\|x-y\|^2}{2\sigma^2}}.
\]

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 in the previous subsections the centers are located at the data points of the training set. Furthermore, the number of centers is here automatically determined.

3 Applications

We present some results achieved by testing the network training strategies in pattern recognition applications.

3.1 High-Resolution ECGs

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
noninvasive marker for an arrhythmogenic substrate and for an increased risk for malignant ventricular tachyarrhythmias. Ventricular late potential analysis (VLP) is herein the generally accepted noninvasive method to identify patients with an increased risk for reentrant ventricular tachycardias and for risk stratification after myocardial infarction [16]. 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. From \( V \) three parameters are extracted: \( QRSd \) the total duration of the QRS, \( tRMS \) the root mean square of the terminal 40 ms of the QRS, and \( LAS \) the terminal low-amplitude signal of the QRS [16]. These measurements are used as features for the conventional analysis and for the RBF classifier.

Here, high-resolution ECGs were obtained from 137 subjects separated into two groups. Group 1 consisted of 66 patients with coronary heart disease in which clinical sustained ventricular tachycardia was inducible at electrophysiologic study. Group 0 consisted of 71 healthy subjects, all with normal resting ECGs.

Results
The classification results are presented in the following way:

<table>
<thead>
<tr>
<th>actual classification</th>
<th>desired classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>RN</td>
</tr>
<tr>
<td>1</td>
<td>FP</td>
</tr>
</tbody>
</table>

The performance of the classifier is evaluated by the measurements sensitivity (SENSI), specificity (SPECI), positive predictive value (PPV), negative predictive value (NPV) and accuracy (ACC). Utilizing the leave-one-out method the RBF-classifier trained by backpropagation training of all network parameters (20 prototypes each with a single scaling parameter; initialization of the centers by LVQ network training) performed as follows:

<table>
<thead>
<tr>
<th>Acc</th>
<th>Sensi</th>
<th>Speci</th>
<th>PPV</th>
<th>NPV</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.854</td>
<td>0.803</td>
<td>0.901</td>
<td>0.883</td>
<td>0.831</td>
</tr>
</tbody>
</table>

The conventional time domain analysis of the high-resolution ECG data gave the following results:

<table>
<thead>
<tr>
<th>Acc</th>
<th>Sensi</th>
<th>Speci</th>
<th>PPV</th>
<th>NPV</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.759</td>
<td>0.682</td>
<td>0.831</td>
<td>0.790</td>
<td>0.740</td>
</tr>
</tbody>
</table>

Here a subject is classified to have VLPs if at least two of the three criteria: \( QRSd > 114 \text{ ms} \), \( tRMS < 20 \mu \text{ V} \), \( LAS > 38 \text{ ms} \) are met [16]. This indicates, that it is possible to attain superior classification results in comparison to conventional methods with general trainable neural network classifiers.

### 3.2 Handwritten Digits

The data set used for evaluating the performance of the classifier consists of 20,000 handwritten digits (2,000 samples per class). The digits, normalized in height and width, are represented through a \( 16 \times 16 \) matrix \( g_{ij} \) where \( g_{ij} \in \{0, \ldots, 255\} \) is a value from a 8 bit gray scale (for details concerning the data set see [17]). The whole data set has been divided into a set of 10,000 training samples and a set of 10,000 test samples. The training set has been used to design the classifiers, and the test set for testing the performance of the classifiers. The results in Table 1 were achieved by several trials. At least three different classifiers per architecture (except the 5NN) were trained and tested. The error rate shown in Table 1 is the median of the measured error rates.

![Handwritten digits](data-set.png)

Figure 3: Handwritten digits from the data set.

For this data set we give results for the following classifiers and training procedures:

- **5NN:** 5-nearest neighbour classifier.
- **LVQ:** 1-nearest neighbour classifier trained by Kohonen’s software package OLVQ1 and LVQ3 training; 50 training epoches; 500 prototypes.
- **RBF-kmeans:** Unsupervised training of the prototypes; a single scaling parameter per basis function set to the mean of the 5 closest prototypes; output layer training through pseudo inverse matrix solution; 50 prototypes; 50 training epoches.
- **RBF-LVQ:** LVQ1 training of the prototypes; one scaling parameter per basis function set to the mean of the 5 closest prototypes; output layer training through pseudo inverse matrix solution; 500 prototypes = 50 per class; 50 LVQ training epoches.
- **RBF-C4.5:** Decision tree training utilizing Quinlan’s software package C4.5 using the splitting parameter \( m \) equal to 1 in order to generate regions with sample size \( \geq 1 \). This leads to an RBF network with 505 RBF centers. Output layer training through the pseudo inverse matrix
Table 1: Results for the handwritten digits.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>error [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>5NN</td>
<td>2.34</td>
</tr>
<tr>
<td>LVQ</td>
<td>3.01</td>
</tr>
<tr>
<td>RBF-kmeans</td>
<td>4.14</td>
</tr>
<tr>
<td>RBF-LVQ</td>
<td>3.06</td>
</tr>
<tr>
<td>RBF-C4.5</td>
<td>5.62</td>
</tr>
<tr>
<td>RBF</td>
<td>1.51</td>
</tr>
<tr>
<td>SVM</td>
<td>1.40</td>
</tr>
<tr>
<td>MLP</td>
<td>2.41</td>
</tr>
<tr>
<td>Poly40</td>
<td>1.66</td>
</tr>
<tr>
<td>RBF40</td>
<td>1.55</td>
</tr>
<tr>
<td>SVM40</td>
<td>1.44</td>
</tr>
</tbody>
</table>

well, particularly in comparison to RBF classifiers trained in two stages. The performance of RBF classifiers can significantly be improved by additional gradient descent learning in order to fine-tune all network parameters simultaneously. The RBF classifier performs as well as other regression-based methods like MLPs or polynomials. This is not surprising, as RBFs, MLPs and polynomials are approximation schemes dense in the space of continuous functions. The SVM and SVM40 classifiers perform very well in our experiments. The error rates for SVM, SVM40, Poly40, RBF, and RBF40 are very similar. Although the performances of the SVM and RBF classifiers are approximately identical, the architectures are completely different. The complete SVM classifier architecture consisting ten classifiers with approximately 4,200 support vectors selected from the finite training data set, whereas the RBF classifier has a hidden layer containing 200 representative prototypes of the input space.

3.3 3-D Visual Object recognition

The recognition of 3-D objects from 2-D 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 [18, 19], artificial neural networks [20, 21, 22], computer vision and pattern recognition [23, 24, 25, 26, 27], psychophysics and brain theory [28, 29, 30]. Due to the increasing performance of current computer systems and the increasing development of computer vision and pattern recognition techniques several 3-D object recognition systems have been developed [31, 32, 33, 34]. The recognition of an 3-D object consisted the following three subtasks (details may be found in [35]):

1. Localization of objects in the camera image.

   In this processing step the entire camera image is segmented into regions. Each region should contain exactly one single 3-D object. Only these marked regions, which we call the regions of interest (ROI), are used for
the further image processing steps. Colour-based approaches for the ROI-detection are used.

Figure 5: Elements of the feature extraction method. The grey valued image (left) is convolved with the masks $S_r$ and $S_b$ resulting in the gradient image (center; absolute value of the gradient). Orientation histograms (right) of non-overlapping subimages constitute the feature vector.

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 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. 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.

Data

Camera images were recorded for six different 3-D objects (orange juice bottle, small cylinder, large cylinder, cube, ball and bucket) with an initial resolution of 768 × 576 pixels. To these five objects nine classes were assigned (bottle lying/upright, cylinders lying/upright). The test scenes were acquired under mixed natural and artificial lighting. Regions of interest where calculated from 1800 images using the colour blob detection method described in section. These regions where checked and labeled by hand, 1786 images remained for evaluation. Regions of interest are detected using three colour ranges, one for red (bucket, cylinder, ball), blue (cylinder) and yellow (cylinder, bucket, orange juice). The image in Figure 4 gives an example of the automatically extracted region of interest. Features were calculated from concatenated 5 × 5 histograms with 3 × 3 Sobel operator, see Figure 5.

Results

In order to get an overview over the underlying structure of the data, two data analysis tools are briefly mentioned. They are useful to explore large sets of high dimensional feature vectors:

Figure 6: 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).

Figure 7: A 54 × 54 (Euclidean) distance matrix for 54 k-means prototypes (6 prototypes per class) of the recorded camera images. The 54 prototypes are sorted by its class memberships in such a way that the prototypes $c_1, \ldots, c_9$ are members of class 0, $c_{10}, \ldots, c_{12}$ have class label 1, etc. Distances between prototypes have been encoded into gray values (small distances black, large distance white).

1. Clustering of the high dimensional feature vectors utilizing for example the k-means clustering algorithm in order to get a smaller set of representative prototypes. The feature vectors from each class are clustered separately leading to a set of prototypes, each
representing a certain class. For the union of all prototypes a distance matrix is calculated. This distance matrix can be plotted as a matrix of gray values (see Figure 7) and used for further data analysis.

The distance matrix of $6 \times 9 = 54$ k-means prototypes of the recorded camera images is shown in Figure 7. Small distances can be observed between prototypes within the classes $\{2, 3, 4, 5, 6, 8\}$ and within $\{0, 1\}$. The prototypes of class 7 seem to be separated from the others, but some smaller distances to prototypes of classes $\{0, 1, 4, 5, 6\}$ can be detected. These smaller distances between prototypes of different classes typically lead to misclassifications.

![Figure 8: A distance preserving 2-D projection of the 54 prototypes (see text).](image)

2. **Nonlinear distance preserving projections** of the data points into a low-dimensional projection space (typically $\mathbb{R}^2$) may be used to explore the data set. The projection of large data sets is computational expensive and intractable, therefore only the prototypes are projected which gives a rough overview over the data (see [36] for some more details on clustering and visualization of large and high-dimensional data sets). In Figure 8 the 54 prototypes of the k-means cluster procedure are shown as projections (with class labels) to $\mathbb{R}^2$. Here, in the lower left part of Figure 8 the projections of the prototypes of classes 0, 1 and 7 are located, the prototypes of the other classes cover the rest of the projection area. This is a similar observation as already seen for distance matrix of the cluster centers. This result is not surprising because Spearman’s rank order coefficient $r_s$ is within the interval $[-1, 1]$, where larger values close to 1 indicate similar rank orders of the sequences, so that the calculated value of $r_s = 0.98$ shows that the order of the distances between the cluster centers in the feature space are fairly well represented by the distances of the corresponding projections in $\mathbb{R}^2$. For a more detailed visualization of the data set clustering and projection algorithms have to be applied to the feature vectors of the confusion classes.

The classification results of four different classifiers are presented for both data sets:

- **1NN**: 1-nearest neighbour classifier with Euclidean distance.
- **LVQ**: 1-nearest neighbour classifier trained by Kohonen’s software package OLVQ1 and LVQ3 training each algorithm for 50 training epochs; 10 prototypes per class.
- **RBF-SVM**: A set of support vector networks each with Gaussian kernel function; N-against-rest learning/voting strategy; NAG library for optimization.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>accuracy [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1NN</td>
<td>91.04</td>
</tr>
<tr>
<td>LVQ</td>
<td>92.44</td>
</tr>
<tr>
<td>RBF-SVM</td>
<td>93.89</td>
</tr>
</tbody>
</table>

Table 2: Classification results of the camera images for the 1-nearest-neighbour classifier, an LVQ-classifier trained by OLVQ1 and LVQ3 with $k = 90$ prototypes, and an RBF network trained by SV learning. The mean of three 5-fold cross-validation runs is given.

4 Conclusion

In this paper training algorithms for RBF networks have been presented and applied in three different applications: 3-D visual object recognition, classification of high-resolution electrocardiograms, and optical character recognition. In comparison to other techniques it is shown that RBF classifiers can be trained very efficiently with two-stage-training. We presented three different algorithms for the initialization of the first layer of an RBF network, utilizing clustering, vector quantization, and classification tree methods. This first step of RBF learning is closely related to density estimation, in particular when unsupervised clustering methods are used.
Parameters of RBF networks can be interpreted easily. Typically the centers of the RBF-nodes are means of feature vectors that can be considered as special patterns of the feature space. Thus, the RBF centers may be analyzed or visualized in the same way as the data. This is an interesting property of RBF networks which may be important in applications where a classifier system has to be build from a specialist in the certain domain who is not an expert in the field of classifier design.

References


