Initialisation of Radial Basis Function Networks Using Classification Trees

Friedhelm Schwenker Christian Dietrich

University of Ulm, D-89069 Ulm, Germany
Email: schwenker, chris@neuro.informatik.uni-ulm.de

Abstract

Learning in radial basis function (RBF) networks is the topic of this paper. Particularly we address the problem of intialisation the centers and scaling parameters in RBF networks utilizing classification tree algorithms. This method was introduced by Kubat in 1998. Algorithms for the calculation of the centers and scaling parameters in an RBF network are presented and numerical results for these algorithms are shown for two different data sets.

1 Introduction

Classification tree induction was introduced by Hunt in 1966 [3]. Classification trees partition the feature space into pairwise disjoint regions. The binary classification tree is the most popular type. Here each node has either two or zero children. Each node in the classification tree is representing a region in the feature space. Typically, these regions are hyperrectangles parallel to the axes of the feature space. One of the most popular decision tree algorithm is Quinlan's C4.5 [10].
The RBF network model is motivated by the locally tuned response observed in biologic neurons. Neurons with a locally tuned response charateristic 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 feature space. RBF networks were introduced into the neural network literature by Broomhead and Lowe in 1988 [1]. In the learning phase of a standard neural network model like a multilayer perceptron all parameters are usually adapted simultaneously at the same time by a global optimization procedure called backpropagation. In contrast, learning in an RBF network may be done in two stages:

1. Calculating the centers and the scaling parameters of the RBFs.
2. Calculation of the output weights.

To determine the centers of the RBFs typically unsupervised or supervised training procedures from clustering or vector quantization are used [8, 11] and the output weights were determined using the pseudo-inverse matrix approach.
Initialising RBF networks using decision trees was introduced by Kubat in 1998 [6]. He was the first who suggested such a combination; Orr addressed this topic in [9]. In this initialisation technique at first a decision tree is calculated using for example the C4.5 software package. From this decision tree the regions defined by the leaves of the tree are transformed into the hidden layer nodes of an RBF network.
The paper is organized as follows: In section 2 we introduce RBF networks; section 3 deals with classification trees. Learning algorithms for RBF networks, including the topics of initialisation the centers and scaling parameters of RBFs, training the output weight matrix and backpropagation learning, are presented in section 4. In section 5 we describe and discuss some results achieved with the discussed algorithms on different benchmark data sets.

2 RBF-Networks

The theoretical basis of the RBF approach lies in the field of interpolation of multivariate functions $f : \mathbb{R}^d \rightarrow \mathbb{R}^k$. The goal of interpolating a set of tupels $(x^\mu, y^\mu)_{\mu=1}^M$ is to find a function F with $F(x^\mu) = y^\mu$ for all $\mu = 1, \ldots, M$, where F is a linear combination of basis functions. The most popular and widely used RBF is the Gaussian function

$$\varphi_j(x) = exp(-\frac{||x - \mu_j||^2}{2\sigma_j^2})$$

where $||\cdot||$ denotes the Euclidean norm, x is a vector of $\mathbb{R}^d$, $\mu_j \in \mathbb{R}^d$ a so-called center or prototype vector and $\sigma_j \in \mathbb{R}$ a scaling parameter.
An RBF network which classifies an input feature vector x^μ into one of k different classes consists of d input neurons, k output neurons and a layer of nonlinear hidden RBF neurons. For the classification task, the target vector $y^\mu \in \{-1, 1\}^k$ is encoded as a binary vector of length k with exactly a single one. The index l with $y_l^\mu = 1$ indicates that the input feature vector x^μ should be categorized to class l.

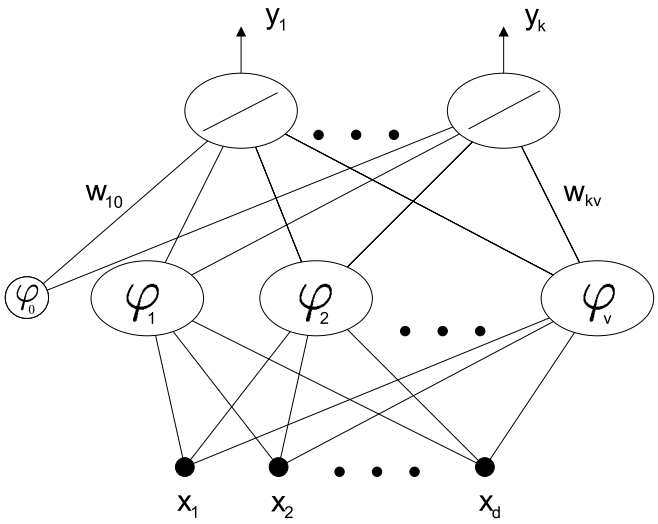


Figure 1: Radial basis function network with d input neurons, v RBF neurons in the hidden layer, and k output neurons.

The output of the j-th RBF neuron is the value of the RBF $\varphi_j(x)$ (see Figure 1). The value of the p-th output neuron is given as a linear combination of the RBFs ($\varphi_0(x) = 1$)

$$y_p = \sum_{j=0}^{v} w_{jp}\varphi_j(x)$$

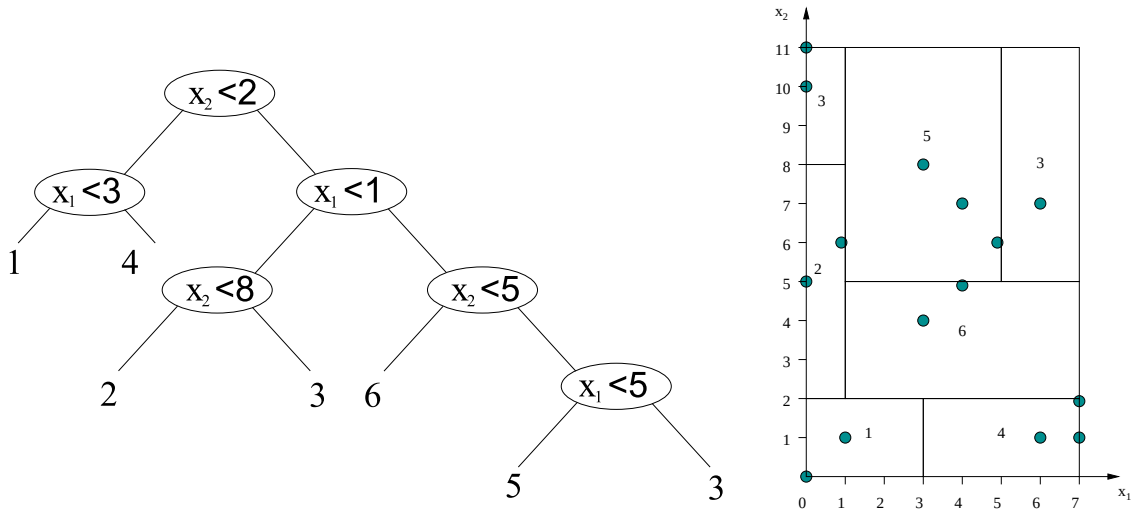
here w_{jp} is the weight from the j-th hidden neuron to the p-th output neuron. The index of the output neuron with the maximal output specifies the class determined by the RBF network for the input x^μ.
In practice vector or matrix valued scaling parameters σ_j may be used. For example, an RBF neuron with Gaussian function can be defined with scaling parameters $\sigma_j = (\sigma_{1j}, ..., \sigma_{dj})$ in the following way

$$\varphi_j(x) = exp(-\sum_{l=1}^{d} \frac{(x_l - \mu_{lj})^2}{2\sigma_{lj}^2})$$

here each σ_{ij} describes the shape of the Gaussian function of the i-th feature dimension.

3 Classification Trees

Classification trees partition the feature space $\mathbb{R}^d$ into pairwise disjoint regions $R_j, j = \{1, ..., v\}$. The binary classification tree is the most often used type in practice. Here each node has either two or zero children nodes. Each node in a classification tree is representing a region R of $\mathbb{R}^d$. If a node R has exactly two children then the regions represented by the children nodes R_l and R_r have the following properties: $R = R_l \bigcup R_r$ and $R_l \bigcap R_r = \emptyset$. Typically, these regions are hyperrectangles parallel to the axes of the feature space. During the training phase the splitting points are determined. Features carrying the most information about the output tend to be splitted earliest and most often. This is some kind of automatic relevance determination which is a natural feature of classification trees [9]. We use Quinlan's C4.5 software package to generate the classification trees [10].



(a) Decision tree with the classes at the leaves. Nodes within the tree are labelled with the splitting points.

(b) Examples and regions (with class labels) of the $\mathbb{R}^2$ determined through the leaves of the decision tree.

Figure 2: Decision tree with classes at the leaves and the corresponding partition of the feature space.

In Figure 2b a data set of $M = 15$ examples is shown. These points are used to create the classification tree in Figure 2a. Figure 2b shows the regions which were defined by the leaves of the classification tree. 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 $i \in \{1, ..., d\}$ and a boundary $b_i \in \mathbb{R}$. Typically, a class is represented in more than one leaf of the tree; for example class 3 in Figure 2 is represented in two different regions. A region in Figure 2b 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. In the C4.5 software the size of the tree (model complexity) can be controlled by a pruning parameter (`c`) [6, 9]. The parameter `m` does something similar. It specifies the number of data points to create a region and helps to prevent the influence of noisy feature vectors in huge data sets.

4 RBF-Network Training

In multilayer perceptron all parameters are usually trained simultaneously at the same time optimising an error function by gradient descent or related optimisation schemes. Learning in an RBF network may be done in two separate stages:

1. Calculating the parameters of the RBFs, including the centers μ_j and the scaling parameters σ_j

2. Calculation of the output weights w_{jp}.

To determine the centers of the RBFs typically unsupervised or supervised training procedures from clustering or vector quantization are used [8, 11]. The output weight matrix can be calculated through a pseudo inverse matrix solution. In the following the initialisation of the RBF centers and scaling parameters from a classification tree is discussed.

4.1 Initialisation the RBF Centers and Scaling Parameters by Decision Trees

In a classification tree a single class is represented by the union of l disjoint regions $R_1, ..., R_l$, each region is given by one of the leaves. For each region R_j an RBF neuron is specified. We present three different algorithms to initialize the RBF centers μ_j and three algorithms to calculate the scaling parameters σ_j.

RBF Centers

The centers μ_j of the RBFs can be calculated in the following ways:

C1 For each region $R_j = [a_{1j}, b_{1j}] \times \cdots \times [a_{dj}, b_{dj}]$ an RBF center $\mu_j = (\mu_{1j}, \ldots, \mu_{dj})$ is determined through

$$\mu_{ij} = (a_{ij} + b_{ij})/2 \tag{1}$$

for $i = 1, \ldots, d$ (see Figure 3a).

C2 The RBF centers μ_j are placed in the middle of the region R_j as in **C1**, except R_j touches the border of a feature dimension i. In this case μ_j is placed at this border (see Figure 3b). In Figure 3b each center touches a border of the feature space.

C3 An RBF center μ_j is determined through the center of gravity of all data points within the region R_j (see Figure 3c). Provided $p_i \in R_j$, $i = 1, ..., n$ then the RBF center μ_j is defined by

$$\mu_j = \frac{\sum_{i=1}^{n} p_i}{n}.$$

Scaling parameters

The scaling parameters σ_j determine how steaply $\varphi_j(x)$ decreases with growing distance from the center μ_j. To calculate the scaling parameters σ_j different algorithms can be used. Note, that for the algorithms **S1** and **S2** σ_j a vector of $\mathbb{R}^d$ and for algorithm **S3** σ_j it is a real number.

S1 This method has to be combined with method **C1** to define the parameters σ_j, $j = \{1, ..., v\}$ in such a way that all basis functions φ_j have the same value at the border of their region (Figure 3a). The size of the hyperrectangle R_j defines the shape of a hyperellipsoid:

$$\sigma_j = \alpha\,((b_{1j} - a_{1j}), \ldots, (b_{dj} - a_{dj})) \tag{2}$$

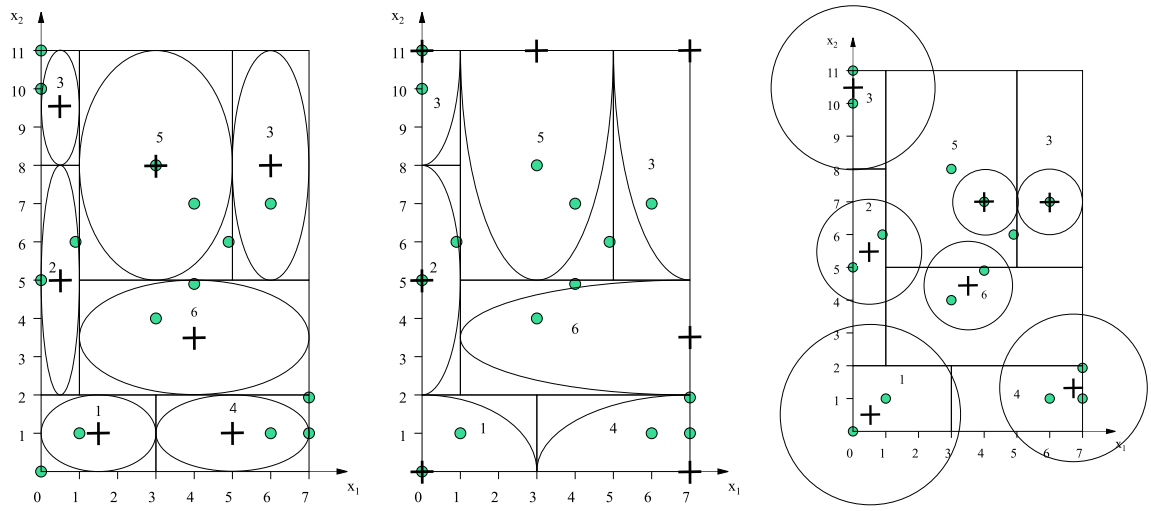
The parameter $\alpha > 0$ specifies the overlap of the RBFs.

S2 This method has to be combined with **C2**. The scaling paramters are defined as in equation 2, except the region touch the border of the feature space then σ_{ij} is multiplied by two (see Figure 3b).

S3 The scaling parameter σ_j is set proportional to the Euclidean distance between μ_j and its nearest neighbour and is calculated through

$$\sigma_j = \min\{\alpha \|\mu_j - \mu_i\| \mid i \neq j\}$$

The factor α defines the overlap of the Gaussians (see Figure 3c, with $\alpha = 0.5$). In principle, this scaling technique may be used with all three algorithms **C1**, **C2** and **C3**.



(a) C1 + S1 (b) C2 + S2 (c) C3 + S3

Figure 3: Regions and data points of our example data set. RBF centers (+)and scaling parameters (shown through the shape of the RBFs) calculated by three different initalisation techniques.

Using the fact that $exp(\sum_l a_l) = \prod_l exp(a_l)$ the RBF can be rewritten as

$$\varphi_j(x) = exp(-\sum_{l=1}^{d} \frac{(x_l - \mu_{lj})^2}{2\sigma_{lj}^2}) = \prod_{l=1}^{d} exp(-\frac{(x_l - \mu_{lj})^2}{2\sigma_{lj}^2}).$$

For irrelevant attributes not appearing as splitting points in the nodes of the decision tree the exponential functions in the product are set to one.

4.2 Output Weight Matrix

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

$$E(W) = \|HW - Y\|^2.$$

The solution is given explicitly in the form $W = H^+ Y$ [12] where H^+ denotes the pseudo inverse matrix of H which is defined as

$$H^+ = \lim_{\alpha \to 0+} (H^T H + \alpha I)^{-1} H^T.$$

Provided that $(H^T H)^{-1}$ is defined, the pseudo inverse matrix becomes simply $H^+ = (H^T H)^{-1} H^T$. After this final step all parameters of the RBF network are determined.

4.3 Backpropagation Training

Backpropagation learning in an RBF-network stands for adapting the output weights W between hidden and output layer, training the prototype vectors $\mu_j \in \mathbb{R}^d$ and the scaling parameters σ_j simultaneously. This can

be achieved by minimizing the least mean square error functional

$$E(\mu, \sigma, W) := \sum_{\mu=1}^{M} \|y^\mu - z^\mu\|^2 \tag{3}$$

for a training set (x^μ, y^μ), $\mu = 1, \ldots, M$ by gradient descent optimization. Here z^μ is the output of the RBF-network. Determining the gradient direction leads to the learning rules for all types of parameters w_p, μ_j and σ_j. For a single example (x^μ, y^μ) of the training set these weight update rules are:

$$\begin{aligned} \Delta w_{jp} &= \eta_t y_j^\mu (y_p^\mu - z_p^\mu) \\ \Delta \mu_{ij} &= \eta_t (x_i^\mu - \mu_{ij})(-\varphi_j'(x^\mu)) \sum_p (y_p^\mu - z_p^\mu) w_{jp}. \end{aligned}$$

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 σ_j becomes

$$\Delta \sigma_j = \eta_t \frac{(r_j^\mu)^2}{\sigma_j^3} \varphi_j'(x^\mu) \sum_p (y_p^\mu - z_p^\mu) w_{jp}.$$

5 Numerical evaluation

We present some results achieved by testing the RBF network training strategies in two applications: Classification of handwritten digits and highresolution electrocardiograms. The main characteristics (number of attributes, number of classes, and number of examples) of these data set are shown in Table 1.

name	*attributes*	*classes*	*examples*
DIGITS	256	10	20,000
IRIS	4	3	150
ECG	3	2	95

Table 1: Main characteristics (attributes, classes, size of data set) of the data sets used in this study.

5.1 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 16×16 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 [5, 7]).

Figure 4: Exampels of the handwritten digits.

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 performance of the classifiers for the handwritten data set is demonstrated in Table 2. Accuracies are shown for the decision tree trained by C4.5 and the RBF network initalise by different transformation techniques. For all CT-RBF networks the output weight matrix was calculated by the pseudo inverse matrix solution. The classification results were determined on the test set.

	CT	CT-RBF
C1 & S1	0.908	0.940
C2 & S2	0.908	0.944
C3 & S3	0.908	0.951
C1 & S3	0.908	0.968

Table 2: Results for the test set of the handwritten digits. CT denotes the decision tree trained by C4.5; CT-RBF denotes the RBF network initialised by a decision tree and output weight matrix calculated through pseudo inverse matrix solution.

5.2 Iris Data Set

The Iris data set dates back to the work of the statistician R.A. Fisher in the early 1980s. It is one of the most famous data sets used in statistics, clustering, and machine learning. It contains fifty examples each of three types of plants each exemplar is described though four numeric attributes.

	CT	CT-RBF	CT-RBF & BP
C1 & S1	0.931	0.921	0.954
C2 & S2	0.931	0.938	0.951
C3 & S3	0.931	0.825	0.947
C1 & S3	0.931	0.882	0.950

Table 3: Classification results for the IRIS data set. Shown is the mean of five 10-fold crossvalidation runs on this data set. CT denotes the results for the decision tree trained by the C4.5 software; CT-RBF denotes the results for the RBF network, initialise by a decision tree and determining the output weight matrix through pseudo inverse matrix solution; CT-RBF & BP means that the CT-RBF network is fine tuned by a backpropagation learning phase.

The performance of the RBF network on the Iris data set is demonstrated in Table 3. Accuracies are shown for decision trees trained by C4.5, RBF networks initalise by different transformation techniques with and output weight matrix calculation by the pseudo inverse matrix solution, and RBF networks initalised by a CT-RBF network which is fine tuned by additional backpropagation training. The results were achieved by 10-fold crossvalidation, here the mean of five different runs are shown.

5.3 High-Resolution ECGs

High-resolution electrocariograms were obtained from 95 subjects separated into two groups. Group 1 consisted patients with coronary heart disease and group 0 consisted healthy subjects, all with normal resting ECGs (see [4] for a full description of the patient group and for highresolution ECG methods).
From the high-resolution ECGs 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 [2]. These measurements are used as input features for the RBF classifier.
The performance of the RBF network for the ECG data set is demonstrated in Table 4. Results were achieved as for the Iris data.

6 Conclusion

We discussed learning in RBF networks and presented algorithms for the initialisation of centers and scaling parameters in RBF networks utilizing classification trees. For the construction of the classification trees the

	CT	CT-RBF	CT-RBF & BP
C1 & S1	0.832	0.863	0.867
C2 & S2	0.832	0.861	0.861
C3 & S3	0.832	0.854	0.858
C1 & S3	0.832	0.863	0.866

Table 4: Classification results for the ECG data set (mean of five 10-fold crossvalidation runs is shown). CT, CT-RBF and CT-RBF & BP as in Table 3.

C4.5 package was used. For the large and highdimensional data set of handwritten digits it can be observed all CT-RBF networks outperform the C4.5 decision tree classifier significantly; a significant difference between the transformation techniques cannot be detected. This is not the case for the IRIS data set. For this data set we find significant differences between the intialisation methods. But these disappear after the RBF networks were fine tuned through a backpropagation learning phase. For the small and low dimensional ECG data set we can observe the all RBF networks show approximately the same performance which is better than the result for the classsification tree.
In the final version of this paper we will present results on some more benchmark data sets, and discuss the topic of fine tuning the CT-RBF-networks through an additional optimisation phase in order to improve the performance of the RBF classifier.

References

[1] D.S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

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

[3] E.B. Hunt, J. Marin, and P.J. Stone. *Experiments in Induction.* Academic Press, 1966.

[4] H.A. Kestler, J. Wöhrle, and M. Höher. Cardiac vulnerability assesment from electrical microvariability of high-resolution electrocardiogram. *Medical and Biological Engineering and Computing*, 38:88–92, 2000.

[5] U. Kreßel. The Impact of the Learning-Set Size in Handwritten-Digit Recognition. In T. Kohonen, editor, *Artificial Neural Networks.* ICANN-91, North-Holland, 1991.

[6] M. Kubat. Decision trees can initialize radial-basis-function networks. *IEEE Trans. Neural Networks*, 9:813–821, 1998.

[7] D. Michie, D.J. Spiegelhalter, and C.C. Taylor. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, 1994.

[8] J. Moody and C. J. Darken. Fast Larning in Networks of locally-tuned Processing Units. *Neural Computation*, 1:284–294, 1989.

[9] M.J.L. Orr. Combining Regression Trees and Radial Basis Function Networks. 1 1, Institute for Adaptive and Neuronal Computation, Devision Of Informatiks, Edinburgh University, 1999.

[10] J.R. Quinlan. *C4.5 Programs for Machine Learning.* Morgan Kaufmann, 1992.

[11] F. Schwenker, H. A. Kestler, G. Palm, and M. Höher. Similarities of LVQ and RBF learning. In *Proc. IEEE Int. Conf. SMC*, pages 646–651, 1994.

[12] F. Schwenker, H.A. Kestler, and G. Palm. Supervised and Unsupervised Learning in Radial Basis Function Networks. NC 2000 (accepted for publication), 2000.