Neural Processing Letters **18:** 139–154, 2003.

On the Kernel Widths in Radial-Basis Function Networks

NABIL BENOUDJIT and MICHEL VERLEYSEN
Université Catholique de Louvain, Microelectronics Laboratoy, Place du Levant 3, B-1348 Louvain-la-Neuve, Belgium. e-mail: {benoudjit, verleysen}@dice.ucl.ac.be

Abstract. RBFN (Radial-Basis Function Networks) represent an attractive alternative to other neural network models. Their learning is usually split into an unsupervised part, where center and widths of the basis functions are set, and a linear supervised part for weight computation. Although available literature on RBFN learning widely covers how basis function centers and weights must be set, little effort has been devoted to the learning of basis function widths. This paper addresses this topic: it shows the importance of a proper choice of basis function widths, and how inadequate values can dramatically influence the approximation performances of the RBFN. It also suggests a one-dimensional searching procedure as a compromise between an exhaustive search on all basis function widths, and a non-optimal a priori choice.

Key words. clusters, Gaussian kernels, radial basis function networks, width scaling factor

1. Introduction

Artificial Neural Networks (ANN) are largely used in applications involving classification or function approximation. It has been proved that several classes of ANN such as Multilayer Perceptron (MLP) and Radial-Basis Function Networks (RBFN) are universal function approximators [1–3]. Therefore, they are widely used for function approximation [3].

Radial-Basis Function Networks and Multilayer Perceptrons can be used for a wide range of applications primarily because they can approximate any function under mild conditions; however, the training of RBFN reveals faster than the training of multilayer perceptrons [4]. This fast learning speed comes from the fact that RBFN has just two layers (Figure 1) of parameters (centers+widths and weights) and each layer can be determined sequentially. This paper deals with the training of RBF networks.

MLP are trained by supervised techniques: the weights are computed by minimizing a non-linear cost function. On the contrary the training of RBF networks can be split into an unsupervised part and a linear supervised part. Unsupervised updating techniques are straightforward and relatively fast. Moreover, the supervised part of the learning consists in solving a linear problem, which is therefore also fast, with the additional benefit of avoiding the problem of local minima usually encountered

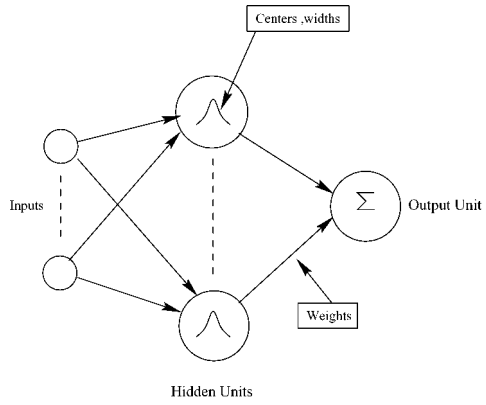

Figure 1. Architecture of a radial basis function network with scalar output.

when using multilayer perceptrons [5]. The training procedure for RBFN can be decomposed naturally into three distinct stages: (i) RBF centers are determined by some unsupervised/clustering techniques, (ii) widths of the Gaussian kernels that are the subject of this paper are optimized, (iii) the network weights between the radial functions layer and the output layer are calculated.

Several algorithms and heuristics are available in the literature regarding the computation of the centers of the radial functions [6–8] and the weights [2, 9]. However, very few papers only are dedicated to the optimization of the widths of the Gaussian kernels. In this paper we show first that the problem of fixing the width is not evident (largely data-dependent) and secondly that it certainly depends on the dimension of the input space. When we are in the presence of a small number of samples (which is always the case in large-dimensional spaces), there is no other choice than an exhaustive search by computation in order to optimize the widths of the Gaussian kernels. In this paper we suggest a one-dimensional searching procedure as a compromise between an exhaustive search on all basis function widths, and a non-optimal a priori choice. Note that gradient descent on all RBFN parameters is still possible, but in this case the speed advantage of RBFN learning is at least partially lost.

The paper is organized as follows. Section 2 reviews the basic principles of a RBFN and presents the width optimization procedure. Section 3 presents simulations performed on simple examples. The simulation results obtained for different dimensions of the input space and a comparison of our approach to three commonly accepted rules are given in Section 4. Section 5 concludes the paper.

2. Radial Basis Function Network

A RBF network is a two-layers ANN. Consider an unknown function $f(\mathbf{x})$: $\Re^d \rightarrow \Re$. In a regression context, RBF networks approximate $f(\mathbf{x})$ by a weighted sum of d-dimensional radial activation functions (plus linear and independent terms, see

below). The radial basis functions are centred on well-positioned data points, called centroids; the centroids can be regarded as the nodes of the hidden layer. The positions of the centroids and the widths of the radial basis functions are obtained by an unsupervised learning rule. The weights of the output layer are calculated by a supervised process using pseudo-inverse matrices or singular value decomposition (SVD) [2]. However, it should be noted that other authors use the gradient descent algorithm to optimize the parameters of RBF network [10]. The training strategies of 'spherical' RBF networks will be detailed in Subsection 2.1.

Suppose we want to approximate a function $f(\mathbf{x})$ with a set of M radial basis functions $\phi_j(\mathbf{x})$, centred on the centroids $\mathbf{c}_j$ and defined by:

$$\phi_j\colon \Re^d \to \Re\colon \phi_j(\mathbf{x}) = \phi_j(\|\mathbf{x} - \mathbf{c}_j\|), \tag{1}$$

where $\|.\|$ denotes the Euclidean distance, $\mathbf{c}_j \in \Re^d$ and $1 \leqslant j \leqslant M$.

The approximation of the function $f(\mathbf{x})$ may be expressed as a linear combination of the radial basis functions [11]:

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{M} \lambda_j \phi_j(\|\mathbf{x} - \mathbf{c}_j\|) + \sum_{i=1}^{d} a_i x_i + b, \tag{2}$$

where λ_j are weight factors, and a_i, b are the weights for the linear and independent terms respectively.

A typical choice for the radial basis functions is a set of multi-dimensional Gaussian kernels:

$$\phi_j(\|\mathbf{x} - \mathbf{c}_j\|) = \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{c}_j)^T \sum\nolimits_j^{-1} (\mathbf{x} - \mathbf{c}_j)\right), \tag{3}$$

where $\sum_j$ is a covariance matrix.

Three cases can be considered:

1. Full covariance matrices $\sum_j$ are replaced by identical scalar widths $\sigma_j = \sigma$ for all Gaussian kernels. In the literature several authors used this option [3, 12–15].
2. Full covariance matrices $\sum_j$ are replaced by different scalars widths σ_j for each Gaussian kernel j. For example, in [16–18], each scalar width σ_j is estimated independently.
3. Full covariance matrices $\sum_j$ represent the general case. Musavi et al. [19] used the covariance matrix to estimate the width of each Gaussian kernel.

In this paper we limit our discussion to the two first cases (1) and (2). One argument to avoid case (3) is that the number of parameters then grows drastically in Equation 2; applications dealing with a small number of learning data are thus difficult to handle by this method. Note that also procedure (3) is very sensitive to outliers [20]. Of course in other situations the use of full covariance matrices may be

interesting [2, 19]. Statistical algorithms used for the estimation of parameters in mixture modelling (as the EM algorithm) could also be used (see for example [2] for the estimation of centers and widths, and [21]). Nevertheless, the use of EM algorithm is based on the maximization of likelihood, making this algorithm more often used for classification and probability estimation problems. The EM algorithm is also sensitive to outliers [22]. In fact, many authors (see for example [4]) suggest to use solution (2) and the learning strategies described below for simplicity reasons. In the following, we will thus concentrate on the study of so-called 'spherical' RBFN covering cases (1) and (2).

2.1. SPHERICAL RBFN LEARNING STRATEGIES

Once the number and the general shape of the radial basis functions $\phi_j(\mathbf{x})$ are chosen, the RBF network has to be trained properly. Given a training data set T of size N_T,

$$T = \{(\mathbf{x}_p, y_p) \in \Re^d \times \Re, 1 \leqslant p \leqslant N_T : y_p = f(\mathbf{x}_p)\}, \tag{4}$$

the training algorithm consists of finding the parameters $\mathbf{c}_j$, σ_j and λ_j, such that $\hat{f}(\mathbf{x})$ fits the unknown function $f(\mathbf{x})$ as close as possible. This is realised by minimising a cost function (usually the mean square error between $\hat{f}(\mathbf{x})$ and $f(\mathbf{x})$ on the learning points). Often, the training algorithm is decoupled into a three-stage procedure:

- determining the centers $\mathbf{c}_j$ of the Gaussian kernels,
- computing the widths σ_j of the Gaussian kernels,
- computing the weights λ_j and independent terms a_i and b.

Moody and Darken [16] proposed to use the k-means clustering algorithm to find the location of the centroids $\mathbf{c}_j$. Other authors use a stochastic online process (Competitive learning) method [7, 17], which leads to similar results, with the advantage of being adaptive (continuous learning, even with evolving input data).

The computation of the Gaussian function widths σ_j is the subject of this paper; it will be detailed at the end of this section.

Once the basis function parameters are determined, the transformation between the input data and the corresponding outputs of the hidden units is fixed. The network can thus be viewed as an equivalent single-layer network with linear output units. Minimisation of the average mean square error yields the well-known least-square solution for the weights.

$$\lambda = \phi^+ y = (\phi^T \phi)^{-1} \phi^T y, \tag{5}$$

where λ, y are the row vectors of λ_j weight factors and y_p training data outputs (of sizes M and N_T respectively), ϕ is the $N_T \times M$ matrix of $\phi_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{c}_j\|^2 / 2\sigma_j^2)$ values and $\phi^+ = (\phi^T \phi)^{-1} \phi^T$ denotes the pseudo-inverse of ϕ. In practice, to avoid possible numerical difficulties due to an ill-conditioned matrix $\phi^T \phi$, singular value decomposition (SVD) is usually used to find the weights [2].

The second stage of the training process involves the computation of the Gaussian function widths, while fixing the degree of overlapping between the Gaussian kernels. This allows finding a compromise between locality and smoothness of the function $\hat{f}(\mathbf{x})$. We consider here both cases (1) and (2) quoted in Section 2. Case (1) consists in taking identical widths $\sigma_j = \sigma$ for all Gaussian kernels [3, 12–15]. In [15] for example, the widths are fixed as follows:

$$\sigma = \frac{d_{\max}}{\sqrt{2M}}, \tag{6}$$

where M is the number of centroids and $d_{\max}$ is the maximum distance between any pair of them. This choice would be close to the optimal solution if the data were uniformly distributed in the input space, leading to a uniform distribution of centroids. Unfortunately most real-life problems show non-uniform data distributions. The method is thus inadequate in practice and an identical width for all Gaussian kernels should be avoided.

If the distances between the centroids are not equal, it is better to assign a specific width to each Gaussian kernel. For example, it would be reasonable to assign a larger width to centroids are widely separated from each other and a smaller width to closer ones [4]. Case (2) therefore consists of estimating the width of each Gaussian kernel independently. This can be done, for example, by splitting the learning points $\mathbf{x}_p$ into clusters according to the Voronoi region associated to each centroid[1], and then computing the standard deviation σ_j^c of the distance between the learning points in a cluster and the corresponding centroid; in reference [17] for example, it is suggested to use an iterative procedure to estimate this standard deviation. Moody and Darken [16], on the other hand, proposed to compute the width factors σ_j (the radius of kernel j) by the p-nearest neighbours heuristic:

$$\sigma_j = \frac{1}{p}\left(\sum_{i=1}^{p} \|\mathbf{c}_j - \mathbf{c}_i\|^2\right)^{\frac{1}{2}} \tag{7}$$

where the $\mathbf{c}_j$ are the p-nearest neighbours to centroid $\mathbf{c}_i$. A suggested value for p is 2 [16]. Saha and Keeler [18] proposed to compute the width factors σ_j by nearest neighbour heuristic where σ_j (the radius of kernel j) is set to the Euclidean distance between $\mathbf{c}_j$ (the vector determining the centre for the jth RBF) and its nearest neighbour $\mathbf{c}_i$, multiplied by an overlap constant r:

$$\sigma_j = r.\min(\|\mathbf{c}_j - \mathbf{c}_i\|). \tag{8}$$

This second class of methods offers the advantage of taking the distribution variations of the data into account. In practice, they are able to perform much better

[1] A Voronoi region is the part of the space nearest to a specific centroid than to any other one.

than fixed-width methods, as they offer a greater adaptability to the data. Even though, as we will show in Section 4, the width values given by the above rules remain sub-optimal.

2.2. WIDTH SCALING FACTOR OPTIMIZATION

We suggest in this subsection a procedure for the computation of the Gaussian function widths based on an exhaustive search belonging the second class of algorithms quoted in subsection 2.1; the purpose is to show the importance of the optimization of Gaussian widths. Therefore we select the widths in such a way to guarantee a natural overlap between Gaussian kernels, preserving the local properties of RBFN, and at the same time to maximize the generalization ability of the network.

First we compute the standard deviations σ_j^c of the learning data in each cluster in a classical way.

DEFINITION. Sigma_cluster is the empirical standard deviation of the learning data contained in a cluster or Voronoi region associated to a centroid.

Subsequently, we determine a width scaling factor WSF, common to all Gaussian kernels. The widths of the kernels are then defined as:

$$\forall j, \quad \sigma_j = \mathrm{WSF}\sigma_j^c. \tag{9}$$

Although the EM algorithm (for example [22, 23]) could be used to optimize all σ_j simultaneously, it appears that in practical situations it is sometimes difficult to escape from local minima, leading to non-optimal solution. Equation (9) then offers a compromise between the usual methods without optimization of σ_j and a M-dimensional optimization of all σ_j together.

By inserting the width factor WSF, the approximation function $\hat{f}(\mathbf{x})$ is smoothed such that the generalization process is possibly improved, and an optimal overlapping of the Gaussian kernels is allowed. Unfortunately, the optimal width factor WSF depends on the function to approximate, the dimension of the input set, as well as on the data distribution. The choice of the optimal WSF value is thus obtained by extensive simulations (cross-validation): the optimal $\mathrm{WSF}_{\mathrm{opt}}$ value will be chosen as the one minimizing the error criterion (mean square error on a validation set), among a set Q of possible WSF values.

When several minima appear, it is recommended to choose the one corresponding to the smallest width scaling factor. Indeed, large WSF have to be avoided for complexity, reproducibility and/or numerical instability.

3. Simulations

We consider a simple example, i.e. we try to approximate the unity identity function ($y_p = 1$) on a d-dimensional hypercube domain $[0,10]^d$.

It must be mentioned here that this problem is purely theoretical: there is no interest to approximate such a linear (and even constant!) function by a RBFN. If the RBFN model in Equation 2 is used to approximate this function, all weights λ_j multiplying the Gaussian kernels should be equal to zero. In order to reach the goal of this paper, i.e. to have insights about the optimal values of the kernel widths, the linear and constant terms were removed from Equation 2 in the simulations. Nevertheless, the objective of this paper is to evaluate the variances of the Gaussian kernels with respect to the dimension of the input space. In order to avoid the consequences of the other parts of the RBFN training algorithm, we chose to work with a constant target function, in order to remove the influence of its variations from our conclusions. Simulations were performed for different dimensions d, in order to see the influence of the dimension on the results.

For all simulations presented in this paper, the density of learning points is uniform in the d-dimensional input space. For this reason, the traditional vector quantization (VQ) step in the RBFN learning process is skipped; the centroids are attached to the nodes of a square grid in dimension d. The goal of this setting is to eliminate the influence of the VQ results on the simulations. It is well known that the placement of centroids on the nodes of a square grid is not the ideal result of a vectorial quantization, when $d \geqslant 2$. For example, it has been demonstrated that in dimension $d = 2$, an ideal vector quantization on a uniform density gives a result in a hive of bee, and not a result in a square grid, as shown it Figure 2 [7]. Nevertheless, it can be shown through a simple calculation that the quantization error obtained with the square grid (Figure 2a) is only about 4% higher than the one obtained with the ideal results (Figure 2b).

As this ideal result is not known in dimensions greater than 2 the assumption is made that the results obtained by placing the centroids on a square grid is a good approximation of those that would be obtained with a true vector quantization.

Once the centroids are placed on a regular grid, the next subsection shows a theoretical way to calculate the optimal width by considering that all the weights are identical in Equation 2. Next, in Subsection 3.2, the optimal width will be estimated by setting all weights λ_j free and calculated according to Equation 5.

(a) (b)

Figure 2. a- scalar quantization (square grid), b- vector quantization (hive of bee).

3.1. THEORETICAL VALUE OF THE OPTIMAL WIDTH OF THE GAUSSIAN KERNELS

As mentioned above, the centroids are placed on a regular grid, and the function to be approximated is constant ($y = \text{constant}$); therefore it is expected that the weights λ_j in Equation 2 will be identical for all centroids. For a theoretical calculation of the optimal WSF coefficient, we will make this assumption and further suppose that their values will be equal to 1. Then, we calculate by Equation 2 (without linear and constant terms) the theoretical output function of the network, and this for various values of σ_j; again, as the centroids are placed on a regular grid, we will suppose that all σ_j values will be identical. The goal is to find the value of σ_j giving the 'flattest' possible output function $\hat{f}(\mathbf{x})$. This one will not be around 1 (there is no reason, since we chose the λ_j equal to 1), but well around another average value m. Taking $\lambda_j = 1$ does not change anything to the problem: if the λ_j were set to $\lambda_j = 1/m$, we would have found an output function with an average value of 1, which was the initial problem. Nevertheless, the two problems bring obviously the same conclusions regarding the widths σ_j.

Note that, $\sigma^c = \sigma_j^c$ ($\forall j \in [1, \ldots, M]$) being constant over all clusters, it is equivalent by Equation 9 to find a optimal value of σ or a optimal value of WSF. In the following section, we will estimate optimal values of σ, in order to make possible the comparison with other methods from the literature (Section 4).

For each value of σ, to find the mean value m we take simply the mean of the output function $\hat{f}(\mathbf{x})$. To quantify the 'flatness' of the output function, we calculate its standard deviation std_y around the mean value m. It should be mentioned here that, in order to avoid as much as possible the border effects encountered when using RBFN, the mean and the standard deviations of $\hat{f}(\mathbf{x})$ are taken only in the middle of the distribution, i.e. in the $[3.85, 6.05]^d$ compact. For each dimension, the σ giving the flatter function $\hat{f}(\mathbf{x})$ is called sigma_theo.

DEFINITION. Identical Gaussian kernels with unit weights ($\lambda_j = 1$) are summed for various σ values

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{M} \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{2\sigma^2}\right),$$

where M is the number of centroids. Sigma_theo is the σ value corresponding to the smallest standard deviation of $\hat{f}(\mathbf{x})$.

As an example, Figure 3 gives the standard deviation std_y of $\hat{f}(\mathbf{x})$ according to σ in dimension 1 and Figure 4 gives the same result in dimension 2.

3.2. EXPERIMENTAL VALUE OF THE OPTIMAL WIDTH OF THE GAUSSIAN KERNELS

In this second set of simulations, we still consider centroids placed on a regular grid, but without the assumption that all λ_j weights will be identical. On the contrary, all weights are set free and we calculate them according to Equation 5 (using Singular

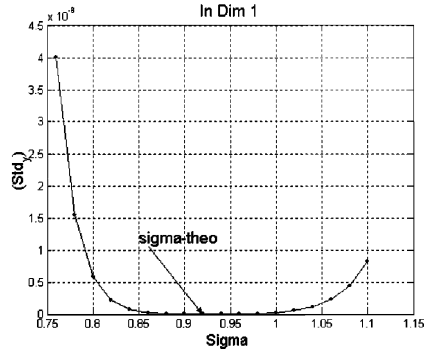

Figure 3. std_y according to σ in dimension 1.

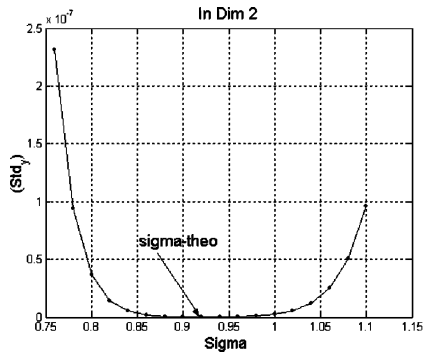

Figure 4. std_y according to σ in dimension 2.

Value Decomposition). As in the previous section, we repeat the experiment for a large set of possible values σ (identical for all Gaussian kernels), and for several dimensions d of the input space. If the principle of 'locality' of the Gaussian Kernels is respected and the border effects are neglected, we should expect identical λ_j. In practice, this is not the case, mainly because of the border effects, as shown it Figure 5a. As in Subsection 3.1, we only used the Gaussian kernels in the centre of the distribution (see Figure 5b) in order to decrease the influence of the border effects.

After the best-fit function is calculated, the performance of the RBF network is estimated by computing an error criterion. Consider a validation data set V, containing N_V data points:

$$V = \left\{(\mathbf{x}_q, y_q) \in \Re^d \times \Re, 1 \leqslant q \leqslant N_V : y_q = f(\mathbf{x}_q)\right\}. \tag{10}$$

The error criterion can be chosen as the mean square error:

$$\mathrm{MSE}_V = \frac{1}{N_V} \sum_{q=1}^{N_V} (y_q - \hat{f}(\mathbf{x}_q))^2, \tag{11}$$

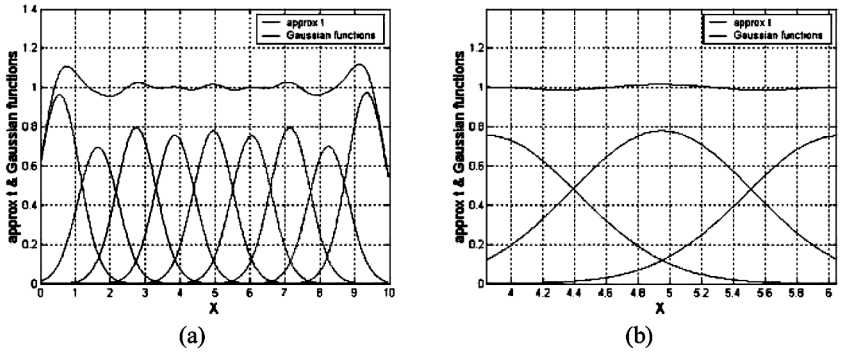

Figure 5. a- with border effects, b- without border effects.

where y_q are the desired outputs. The minimum of the mean square error (MSE_v) gives now another value of sigma, called sigma_exp.

DEFINITION. Identical Gaussian kernels are summed for various σ values

$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{M} \lambda_j \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{2\sigma^2}\right),$$

where M is the number of centroids. Sigma_exp is the σ value corresponding to the smallest MSE_v.

Figure 6(a) gives MSE_v according to σ in dimension 1 and Figure 6(b) gives the same result in dimension 3. Figure 6 shows the presence of two minimums. The first one corresponds to a local decomposition of the function in a sum of Gaussian kernels; this interpretation is consistent with the classical RBF approach. However, the second one corresponds to a non-local decomposition of the function. As a consequence, the weights λ_j turn out to be enormous in absolute value (positive or

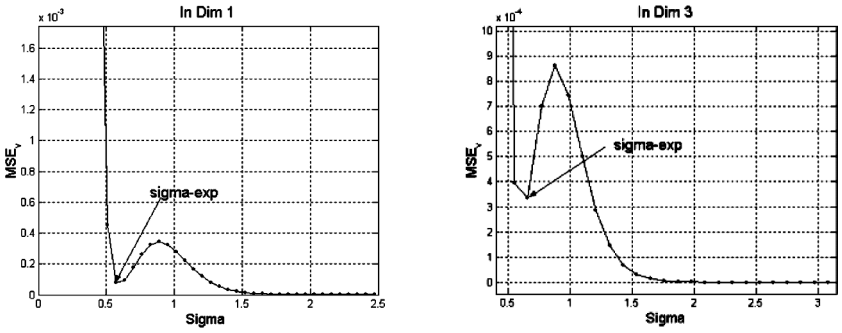

Figure 6. (a) MSE_v versus σ in Dim 1. (b) MSE_v according versus σ in Dim 3.

negative) in order to compensate for the non-flat slopes. This leads to a greater complexity of the RBFN. In addition, large λ_j dramatically increases numerical instability. The optimal value chosen for σ is therefore the one related to the smallest minimum.

4. Results

The simulations were made on databases of points distributed uniformly in a hypercube of edge lengths equal to 10, in various dimensions d. The number of centroids is chosen equal to 9^d. Other simulations made with a different number of centroids gave similar results. The number of training and test points is chosen sufficiently large to avoid (as much as possible) falling into the difficulties due to the empty space phenomenon (between 1000 and 50000 training points according to the dimension of the input space). These restrictions have limited the simulations to a dimension of input space equal to 5.

Table 1 gives the results. In order to obtain values independent from the number of centroids, the 'Width Scaling Factors' WSF_theo and WSF_exp are defined, as being the ratio between sigma_theo and sigma_cluster on one hand, and sigma_exp and sigma_cluster on the other hand respectively. Indeed it is more appropriate to compare results on the scale-independent WSF coefficient instead of the values of σ for two reasons:

- most results in the literature are based on the sigma_cluster value, making the use of WSF easier for comparisons;
- the WSF values are independent from the number of centroids, while those of σ are not.

Several comments result from Table 1:

- sigma_cluster is proportional to the square root of dimension, as shown by a simple analytical calculation:

$$\sigma_{\text{cluster}} = \sqrt{d}\frac{a}{2\sqrt{3}}, \tag{12}$$

Table 1. WSF_theo and WSF_exp according to the dimension of the input space.

Dim	Sigma_cluster	Sigma_theo	Sigma_exp	WSF_theo	WSF_exp
1	0.3175	0.92	0.5715	2.897	1.8
2	0.4490	0.92	0.5837	2.048	1.30
3	0.5499	0.92	0.5506	1.673	1.01
4	0.6350	0.92	0.5676	1.448	0.89
5	0.7099	0.92	0.5471	1.2959	0.77

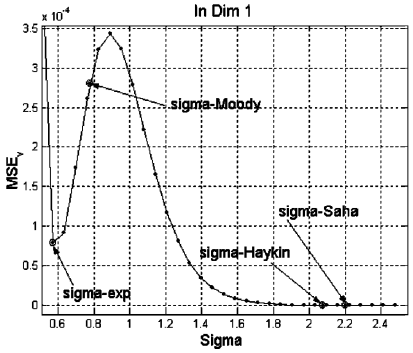

Figure 7. MSE_v according to sigma in Dim 1 with the various calculation methods of σ.

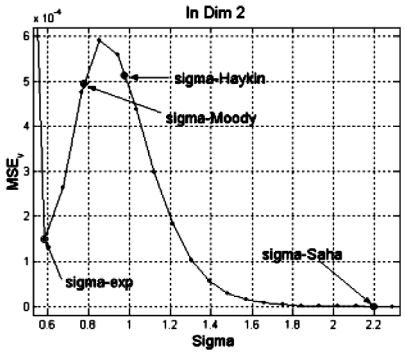

Figure 8. MSE_v according to sigma in Dim 2 with the various calculation methods of σ.

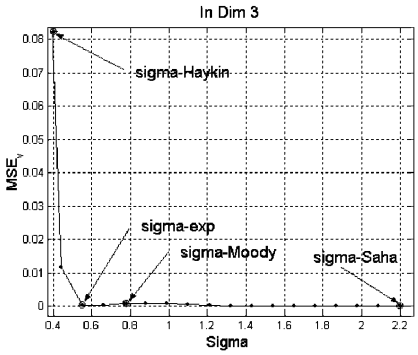

Figure 9. MSE_v according to sigma in Dim 3 with the various calculation methods of σ.

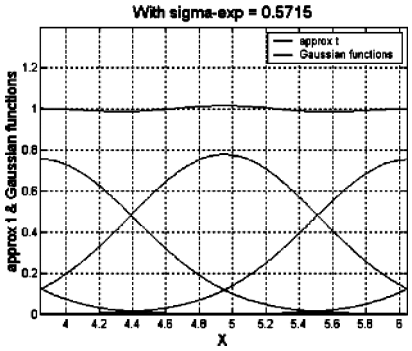

Figure 10. Local decomposition of $y = 1$ *with sigma_exp* $= 0.5715$.

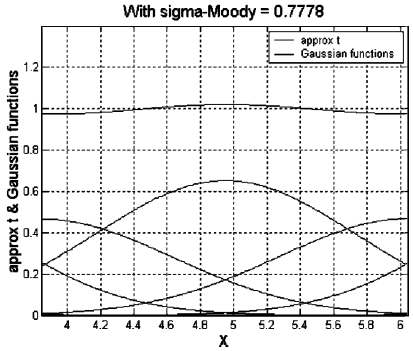

Figure 11. Local decomposition of $y = 1$ *with sigma_Moody* $= 0.7778$.

where a is the length of an edge of the hypercube corresponding to the Voronoi zone of a centroid. In the simulations, the 9^d centroids are placed *a priori* at the positions $0.55 + k*1.1$ (with $1 \leqslant k \leqslant 9$) measured on each axis of input space; a is thus equal to 1.1.

- We notice that sigma_theo does not depend on the dimension of the input space. Therefore, WSF_theo is inversely proportional to the square root of the dimension of the input space.
- We also notice that the sigma_exp values are systematically lower, about 30 to 35%, than the sigma_theo values. This is due to the increased freedom given to the network coefficients by allowing weight variations rather than fixing them.

We also compared our method (Calculation of sigma_exp) to the three approaches of Moody & Darken [16], S. Haykin [15] and A. Saha & J. D. Keeler [18], quoted in Section 2. Figures 7, 8 and 9 illustrate the mean square error obtained according to sigma for different dimension d ($1 \leqslant d \leqslant 3$) with the various calculation methods of

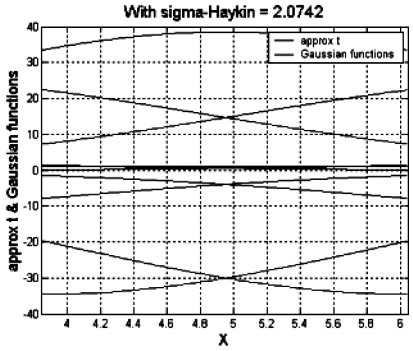

Figure 12. Non-local decomposition of $y = 1$ *with sigma_Haykin* $= 2.0742$.

σ. We notice here that whatever is the dimension of the input space, we always find two minima. Figures 10, 11 and 12 clearly show that the choice of the sigma value has a great influence on the local character of the decomposition in a sum of Gaussian kernels of the function (in dimension 1) to approximate.

5. Conclusion

This paper gives some insights about the choice of Gaussian kernel widths, to use during the training of spherical RBF networks. Indeed, a major part of the literature in the field of RBFN covers the optimization of the positions of Gaussian kernels and the multiplicative weights. On the other hand, the choice of their widths is often based on heuristics, without real theoretical justification.

In this paper, we first show the importance of the choice of the kernel widths. In many situations, a bad choice can lead to an approximation error definitely higher than the optimum, sometimes by several orders of magnitude. Then, we show by two types of simulations, that a classic choice (taking the width of Gaussian kernels equal to the standard deviation of the points in a cluster) is certainly not optimal. For example, in simulations in dimension 1, it appears that the width should be twice this value. It is then suggested to optimize the widths; an example of one-dimensional optimization procedure is presented in this paper, through the use of a multiplying factor to the widths. Finally, we show that the dimension of the data space has an important influence on the choice of σ. In particular, that the multiplicative correction that must be applied to the standard deviation of points in a cluster is shown to be inversely proportional to the square root of the dimension of the input space.

Simulations on real databases (see for example [24]) show, similarly to the curves illustrated in this paper, a strong dependency of the approximation error with respect to the width (and Width Scaling Factor) of the Gaussian kernels. A similar methodology can thus be applied to choose optimum widths, despite the fact that their numerical values depend on the function to approximate.

The results show the need for a greater attention to be given to the optimization of the widths of the Gaussian kernels in RBF networks, and to the development of methods allowing to fix these widths according to the problem without using exhaustive search.

Acknowledgements

Michel Verleysen is Senior Research Associate of the Belgian F.N.R.S. (National Fund For Scientific Research).

References

1. Park, J. and Sandberg, I.: Approximation and radial basis function networks, *Neural Comput*. **5** (1993), 305–316.
2. Bishop, C. M.: *Neural Networks for Pattern Recognition*, Oxford university press, 1995.
3. Park, J. and Sandberg, I. W.: Universal approximation using radial-basis-function networks, *Neural Comput*. **3** (1991), 246–257.
4. Young-Sup Hwang and Sung-Yang Bang.: An efficient method to construct a radial basis function neural network classifier, *Neural Networks*, **10**(8) (1997), 1495–1503.
5. Robert J. Howlett and Lakhmi C. Jain, Radial Basis Function Networks 2: New Advances in Design, Physica-Verlag: Heidelberg, 2001.
6. Ahalt, S. C. and Fowler, J. E.: Vector quantization using artificial neural networks models. In *Proceedings of the International Workshop on Adaptive Methods and Emergent Techniques for Signal Processing and Communications*, June 1993, pp. 42–61.
7. Gresho, A. and Gray, R. M.: *Vector Quanitzation and Signal Compression*, Kluwer International series in engineering and computer science, Norwell, Kluwer Academic Publishers, 1992.
8. David Sanchez, V. A.: Second derivative dependent placement of RBF centers, *Neurocomputing* **7**(3) (1995), 311–317.
9. Omohundro, S. M.: Efficient algorithms with neural networks behavior, *Complex Systems* **1** (1987), 273–347.
10. Verleysen, M. and Hlaváčková, K.: An optimized RBF network for approximation of functions, In: *European Symposium on Artificial Neural Networks (ESANN 94)*, pp. 175–180, Brussels, April 20-21-22, 1994.
11. Tomaso Poggio and Federico Girosi: Networks for approximation and learning, *Proceedings of the IEEE*, **78**(9) (1990), 1481–1497.
12. Orr, M. J.: Introduction to Radial Basis Functions Networks, *Technical reports*, April 1996, www.anc.ed.ac.uk/~mjo/papers/intro.ps.
13. David Sanchez, V. A.: On the number and the distribution of RBF centers, *Neurocomputing* **7**(2) (1995), 197–202.
14. Chen, S. and Billings, S. A.: Neural networks for nonlinear dynamic system modelling and identification, *Int. J. Control*, **56**(2) (1992), 319–346.
15. Haykin, S.: *Neural Networks a Comprehensive Foundation*, Prentice-Hall Inc, second edition, 1999.
16. Moody, J. and Darken, C. J.: Fast learning in networks of locally-tuned processing units, *Neural Comput*. **1** (1989), 281–294.
17. Verleysen, M. and Hlaváčková, K.: Learning in RBF Networks, *International Conference on Neural Networks (ICNN)*, Washington, DC, June 3–9 (1996), pp. 199–204.

18. Saha, A. and Keeler, J. D.: Algorithms for Better Representation and Faster Learning in Radial Basis Function Networks, *Advances in Neural Information Processing Systems 2*, Edited by David S. Touretzky, pp. 482–489, 1989.
19. Musavi, M. T., Ahmed, W., Chan, K. H., Faris, K. B. and Hummels, D. M.: On the Training of radial basis function classifiers, *Neural Networks*, **5** (1992), 595–603.
20. Ripley, B. D.: *Pattern Recognition and Neural Network*, Cambridge University Press, first edition, 1996.
21. Lázaro, M., Santamaría, I. and Pantaleón, C.: A new EM-based training algorithm for RBF networks, *Neural Networks*, **16** (2003), 69–77.
22. Archambeau, C., Lee, J. and Verleysen, M.: On convergence problems of the EM algorithm for finite Gaussian mixtures, In: *European Symposium on Artificial Neural Networks (ESANN'2003)*, pp. 99–104, Bruges, April 23-24-25, 2003.
23. Xu, L. and Jordan, M. I.: On convergence properties of the EM algorithm for Gaussian mixtures, *Neural Computation*, **8**(1) (1996), 129–151.
24. Benoudjit, N., Archambeau, C., Lendasse, A., Lee, J. and Verleysen, M.: Width optimization of the Gaussian kernels in radial basis function networks, In: *European Symposium on Artificial Neural Networks (ESANN'2002)*, pp. 425–432, Bruges, April 24-25-26, 2002.