ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 425-432

Width optimization of the Gaussian kernels in Radial Basis Function Networks

Nabil Benoudjit[1], Cédric Archambeau[1], Amaury Lendasse[2], John Lee[1], Michel Verleysen[1,*]

[1] Université Catholique de Louvain - Microelectronics Laboratoy,
Place du Levant 3, B-1348 Louvain-la-Neuve, Belgium,
Phone : +32-10-47-25-40, Fax : +32-10-47-21-80
Email : {benoudjit, archambeau, lee, verleysen}@dice.ucl.ac.be

[2] Université Catholique de Louvain – CESAME,
Avenue G. Lemaître 4, B-1348 Louvain-la-Neuve, Belgium,
Email : lendasse@auto.ucl.ac.be

Abstract. Radial basis function networks are usually trained according to a three-stage procedure. In the literature, many papers are devoted to the estimation of the position of Gaussian kernels, as well as the computation of the weights. Meanwhile, very few focus on the estimation of the kernel widths. In this paper, first, we develop a heuristic to optimize the widths in order to improve the generalization process. Subsequently, we validate our approach on several theoretical and real-life approximation problems.

1. Introduction

Artificial neural networks (ANN) are largely used in applications involving classification or function approximation. Lately, it has been proved that several classes of ANN are universal function approximators [1]. Therefore, they are widely used for function interpolation [2][3].
Among the ANN classes, we find the radial basis function (RBF) networks and the multi-layer perceptrons (MLP). Both are multi-layered networks and they can be considered as connectionist models. Both need to be trained by a sufficiently large data set to learn the process to approximate. Even though, RBF methods differ from MLP in their training procedure.
MLP are trained by supervised techniques: the set of weights are computed by solving a non-linear constrained equation set. On the contrary the training of RBF networks can be split into an unsupervised part and a supervised but linear part. Unsupervised

[*] Michel Verleysen is Senior Research Associate of the Belgian F.N.R.S. (National Fund For Scientific Research).
Part of this work has been funded by the *European Commission* (IST-2000-25145 “OPTIVIP”) and the *Ministère de la Région wallonne* (“Programme de Formation et d'Impulsion à la Recherche Scientifique et Technologique”).

updating techniques are straightforward and relatively fast. Meanwhile its supervised part consists in solving a linear problem, which is therefore also fast. The training methods used for RBF networks are thus substantially less time and resources consuming [3].

2. Radial basis function network

A RBF network is a three-layered ANN. Consider an unknown function $f(\mathbf{x}) : \Re^d \rightarrow \Re$. In a RBF network, $f(\mathbf{x})$ is approximated by a set of d-dimensional radial activation functions. Those radial basis functions are centred on well-positioned data points, called centroids. The centroids can be regarded as the nodes of the hidden layer. Generally, the position of the centroids and the widths of the radial basis functions are obtained by an unsupervised learning rule, whereas the weights of the output layer are calculated by a supervised, single-shot process using pseudo-inverse matrices or singular value decomposition.

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

$$\phi_j : \Re^d \rightarrow \Re : \phi_j(\mathbf{x}) = \phi_j\left(\left\|\mathbf{x} - \mathbf{c}_j\right\|\right), \qquad (1)$$

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

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

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

where λ_j are weight factors.

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

$$\phi_j\left(\left\|\mathbf{x} - \mathbf{c}_j\right\|\right) = \exp\left(-\frac{1}{2}\left(\frac{\left\|\mathbf{x} - \mathbf{c}_j\right\|}{\sigma_j}\right)^2\right), \qquad (3)$$

where σ_j is the width factor of the jth hidden unit in the hidden layer.

3. Training of a Radial basis function network

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

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

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.

3.1. Error criterion

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:

$$V = \{(\mathbf{x}_q, y_q) \in \Re^d \times \Re, 1 \le q \le N_V : y_q = f(\mathbf{x}_q)\}. \tag{5}$$

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

$$MSE_V = \frac{1}{N_V} \sum_{q=1}^{N_V} \left(y_q - \hat{f}(\mathbf{x}_q) \right)^2 , \tag{6}$$

where y_q are the desired outputs.

3.2. Training algorithm

Often, the training algorithm is decoupled into a three-stage procedure:

1. determine the centres $\mathbf{c}_j$ of the Gaussian kernels,
2. compute the widths of the Gaussian kernels σ_j,
3. compute the weights λ_j.

During the first two stages only the inputs $\mathbf{x}_p$ of the training data set T are used. The parameters are thus adapted according to an unsupervised updating rule. In the third step the weights are updated with respect to the corresponding desired outputs. Meanwhile $\mathbf{c}_j$ and σ_j remain fixed.

In the literature, several algorithms and heuristics are proposed for the computation of the centroids $\mathbf{c}_j$ [4][5] and the weights λ_j [3][6]. The centroids are estimated according to a vector quantization scheme, like for example competitive learning, while the weights are found by solving equation (2). This equation is linear since the radial basis functions $\varphi_j(\mathbf{x})$ are fixed. However, very few papers are dedicated to the optimization of the widths σ_j of the Gaussian kernels.

4. Width factors

Typically two alternatives are considered. The first one consists in taking the widths σ_j equal to a constant for all Gaussian functions [7][8][9]. In [9], for example, the widths are fixed as follows:

$$\sigma = \frac{d_{\max}}{\sqrt{2M}} , \tag{7}$$

where M is the number of centres and d_{max} is the maximum distance between those centres. Such a procedure fixes the degree of overlapping of the Gaussian kernels. It allows to find a compromise between locality and smoothness of the function $\hat{f}(\mathbf{x})$. 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 the 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, since their widths should depend on the position of the centroids, which in turn depend on the data distribution in the input space.

The second option consists in estimating the width of each Gaussian function independently. This can be done, for example, by simply computing the standard deviation of the distance between the data and their corresponding centroid. Reference [10] suggests an iterative procedure to estimate the standard deviation. Moody and Darken [11], on the other hand, proposed to compute the width factors σ_j by the r-nearest neighbours heuristic:

$$\sigma_j = \frac{1}{r}\left(\sum_{i=1}^{r} \left\|\mathbf{c}_i - \mathbf{c}_j\right\|^2\right)^{\frac{1}{2}} \tag{8}$$

where the $\mathbf{c}_i$ are the r-nearest neighbours of centroid $\mathbf{c}_j$. A suggested value for r is 2. 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, as they offer a greater adaptability to the data than a fixed-width procedure. Even though, as we will show next, the widths values remain sub-optimal.

In this paper we propose to unite both approaches. First we compute the standard deviations σ_j^c of each data cluster[1] in a classical way. Subsequently we determine a *width scaling factor q*, common to all Gaussian kernels. The widths of the kernels are then defined as:

$$\forall j, \sigma_j = q\sigma_j^c , \tag{9}$$

By inserting the width scaling factor, the approximation function $\hat{f}(\mathbf{x})$ is smoothed such that the generalization process is more efficient, as we allow an optimal overlapping of the Gaussian kernels.

Unfortunately, the optimal width factor q depends on the function to approximate, the dimension of the input set, as well as on the data distribution. The choice of the optimal width factor is thus obtained by a heuristic.

Consider a width factor set Q. We evaluate, successively, for each value $q_l \in Q$ the error criterion, chosen as the mean square error. The optimal q_{opt} corresponds to the smallest error:

$$\forall l, MSE_V(q_{opt}) \le MSE_V(q_l) . \tag{10}$$

When several minima appear, it is usually recommended to choose the one corresponding to the smallest width scaling factor. Indeed, large q_l have to be avoided for complexity, reproducibility and/or numerical instability. This will be illustrated in the next section, in which we prove the effectiveness of our approach on several artificial and real-life problems.

[1] A cluster is a region associated to each centroid. Such a region is usually called a "Voronoi zone".

ESANN'2002 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 24-26 April 2002, d-side publi., ISBN 2-930307-02-1, pp. 425-432

5. Results

In this section, we show the need to optimise the widths of the Gaussian kernels in order to improve the generalization process, and we compare our heuristic approach to the methods proposed by Moody & Darken [10] and S. Haykin [11].

Consider N_T data selected randomly according to a 1D sine wave:

$$x \in [0,1]: y_1 = \sin(12x)$$

The function and its corresponding approximation are plotted in figure 2. In figure 3, we have plotted the MSE_V in function of q. The experiment was repeated 50 times. One can notice that in some cases a second minimum appears for higher q. Though, the second minimum is rather fortuitous than systematic, as the average curve confirms. In addition, when we improve the learning process by increasing the size of the learning data set T, the non-systematic minimum vanishes (figure 4 and 5). The optimal value for q is thus taken as the first minimum, i.e. approximately 2.

A second data set is given by:

$$x \in [-4,4]: y_2 = 1 + \left(x + 2x^2\right)\sin(-x^2).$$

The function and the RBF approximation are illustrated in figure 6, while its MSE_V is plotted in figure 7. Here again we observe two minima. This time, however, both are systematic, as the average curve shows. Nevertheless, both minima are of a different type.

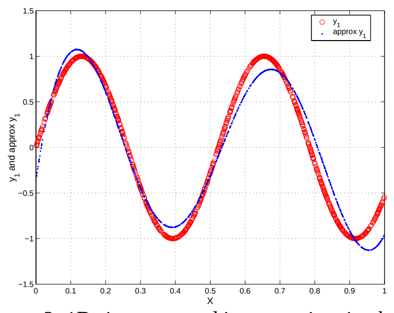

Figure 2: 1D sine wave and its approximation by a RBF network (5 centroids).

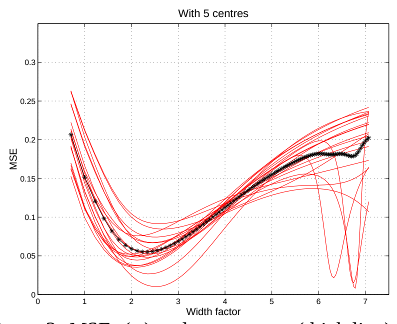

Figure 3: MSE_V (--) and mean curve (thick line) in function of q for the 1D sine wave (N_T = 400).

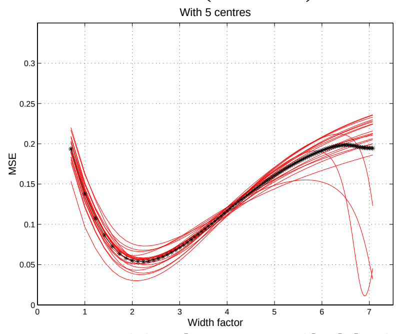

Figure 4: MSE_V (--) and mean curve (thick line) in function of q for the 1D sine wave (N_T = 1400).

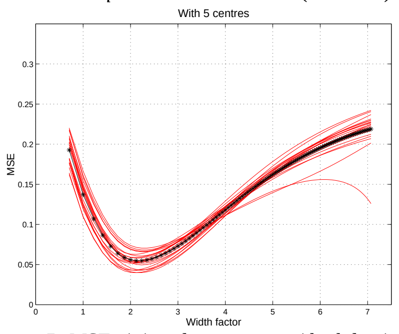

Figure 5: MSE_V (--) and mean curve (thick line) in function of q for the 1D sine wave (N_T = 2000).

The first minimum corresponds to a *local* decomposition of the function in a sum of Gaussian functions (figure 8). This interpretation is consistent with classical RBF network theory.
The second one, on the contrary, corresponds to a non-local decomposition of the function (figure 9). As a consequence, the weights λ_j turn out to be enormous in absolute value (see the 10^5 scale in figure 9) in order to approximate the non-flat slopes. This leads to a greater complexity of the RBFN, which expresses itself by a greater width scaling factor (greater widths). In addition, large λ_j increase numerical instability. Once more, the optimal width factor is the one related with the smaller q.

In table I, we have compared our heuristic approach to the approaches of Moody & Darken [10] and S. Haykin [11], which we quoted in section 4. In both examples our approach exhibits the best compromise between accuracy and complexity. Indeed, we obtain small mean square errors combined with greater locality and small λ_j.

The next three examples are real-life problems, in which ANN were used to model an unknown process. In those problems, the input space is generally multi-dimensional.
The first one aims to determine the water content in a dried milk sample. The training set contains 27 dried milk samples. Each sample contains 7 spectral data representing the input data of the network and the corresponding desired output represents the water content. On the other hand, the validation set contains only 10 dried milk samples.

Figure 10 shows the curve of the mean square error according to the width scaling factor. The choice of the optimal width scaling factor corresponds to the minimum of the mean square error.
The second real-life problem consists of developing a black-box model that predicts the position of phosphenes in the visual field of blinds as reported in [12]. Phosphenes are little lightning spots, dots or stripes appearing in the visual field of blind patients when their optic nerve is electrically stimulated. When we observe the mean square error, we find an optimal width scaling factor for small q (figure 11).
Finally, RBF networks can be used for time series prediction. The principle consists in predicting the next value in the sequence, as a function of the previous values. A well-known time example is the SantaFe A [13]. In this sequence the last six values are used to predict the new one. In figure 12, one can notice that a minimal MSE_V is obtained for a value of 13 of the width scaling factor. It should be mentioned, that, in this case, no local decomposition of the function seems to appear. Indeed, the optimal q is at a high value. As a consequence, numerical instability can already be observed.

	Method	***MSE$_v$***	***Locality***
y_1	Moody and Darken	0.0844	High
	S. Haykin	0.1334	Medium
	heuristic	**0.0533**	High
y_2	Moody and Darken	24.7994	High
	S. Haykin	5.3929	Low
	heuristic	**18.8417**	High

Table I : Comparison of the performances of our heuristic and classical approaches.

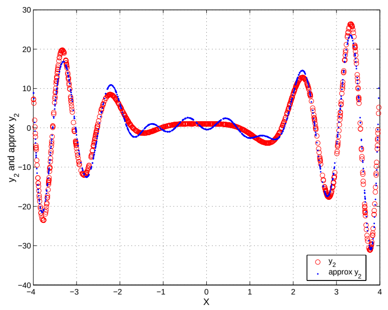

Figure 6: Function y_2 and its approximation by a RBF network (20 centroids).

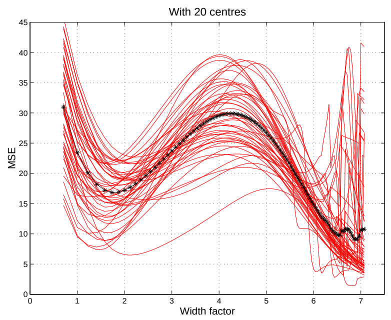

Figure 7: MSE_V (--) and mean curve (thick line) in function of q for y_2.

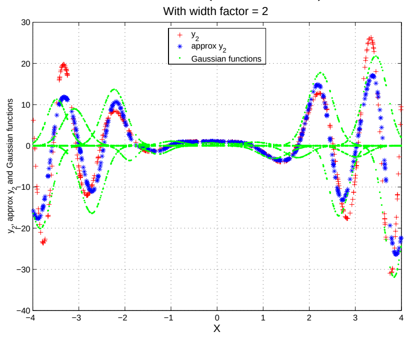

Figure 8: Local decomposition of y_2.

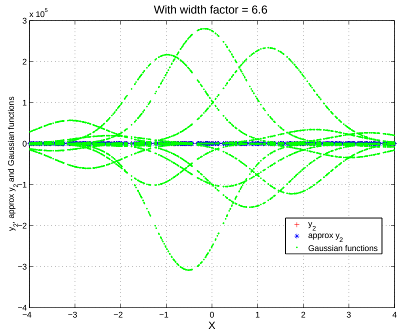

Figure 9: Non-local decomposition of y_2.

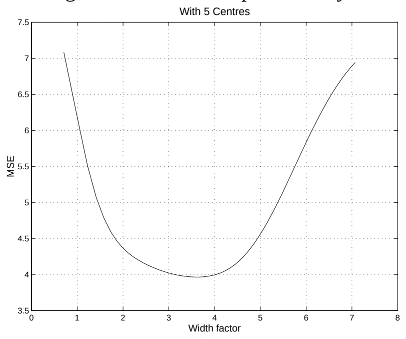

Figure 10: Prediction of the water content in the dried milk.

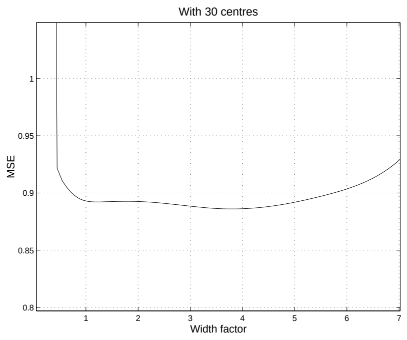

Figure 11: Phosphene prediction.

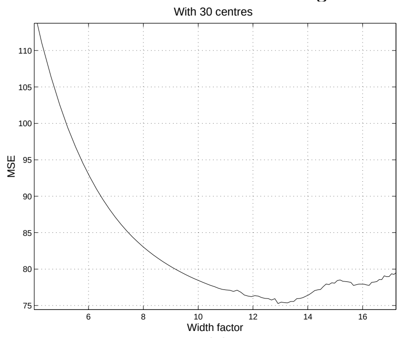

Figure 12: Prediction of financial time series.

6. Conclusion

In this paper we have proposed a heuristic, which allows optimizing the widths of the Gaussian kernels in RBF networks. First we compute the standard deviation of each data cluster. Subsequently we determine a common width scaling factor for the Gaussian kernels. The choice of the optimal width scaling factor corresponds to the smallest mean square error. When several minima appear, it is usually recommended to choose the one corresponding to the smallest width scaling factor in order to avoid instability. The results obtained by using this technique show that fixing the width of the Gaussian kernels a priori or simply by computing the standard deviation of the clusters is sub-optimal.

References

[1] J.Park. and I. Sandberg, *"Approximation and Radial-Basis function Networks"*, Neural Computation, vol. 5, pp. 305-316, 1993.

[2] D. S. Broomhead and D. Lowe, *"Multivariable functional interpolation and adaptive networks"*, Complex Systems 2, pp. 321-355, 1988.

[3] C. M. Bishop, *"Neural Networks for Pattern Recognition"*, Oxford university press, 1995.

[4] S. C. Ahalt and J. E. Fowler, *"Vector Quantization using Artificial Neural Networks Models"*, Proceedings of the International Workshop on Adaptive Methods and Emergent Techniques for Signal Processing and Communications, pp. 42-61, June 1993.

[5] A. Gresho and R. M. Gray, *"Vector Quanitzation and Signal Compression"*, Kluwer International series in engineering and computer science, Norwell, Kluwer Academic Publishers, 1992.

[6] S. M. Omohundro, *"Efficient algorithms with neural networks behaviour"*, Complex Systems 1 pp. 273-347, 1987.

[7] M. J. Orr, *"Introduction to Radial Basis Functions Networks"*, www.anc.ed.ac.uk/~mjo/papers/intro.ps, April 1996.

[8] J. Park. and I. Sandberg, *"Universal Approximation Using Radial-Basis function Networks"*, Neural Computation, vol. 3, pp. 246-257, 1991.

[9] S. Haykin, *"Neural Networks a Comprehensive Foundation"*, Prentice-Hall Inc, second edition, 1999.

[10] M. Verleysen and K. Hlavackova, *"Learning in RBF Networks"*, International Conference on Neural Networks (ICNN), Washington, DC, pp. 199-204, June 3-9 1996.

[11] J. Moody and C. J. Darken, *"Fast learning in networks of locally-tuned processing units"*, Neural Computation 1, pp. 281-294, 1989.

[12] C. Archambeau, A. Lendasse, C. Trullemans, C. Veraart, J. Delbeke, M. Verleysen, *"Phosphene evaluation in a visual prosthesis with artificial neural networks"*, Eunite 2001: european symposium on intelligent technologies, hybrid systems and their implementation on smart adaptive systems, 13-14 December 2001.

[13] A. S. Weigend and N.A. Gershenfeld, *"Times Series Prediction: Forcasting the future and Understanding the Past"*, Addison-Wesley Publishing Company, 1994.