RADIAL BASIS FUNCTION NETWORKS

Kateřina Hlaváčková, Roman Neruda [1]

Abstract: An overview of feedforward networks with one hidden layer with Radial Basis Function (RBF) units is presented. The learning process of this type of network can take advantage of other known algorithms such as clustering, Kohonen maps or regularization. The approximation capabilities of RBF networks are compared with those of the multi-layer perceptron type.

1. Introduction

Radial basis functions have been studied since the middle of eighties. Earlier works (for example [3]) were devoted to the mathematical analysis of that class of functions from the point of view of interpolation and approximation. A detailed mathematical theory can be found in [22].

With respect to neural network theory, the methods using radial basis functions have been applied in approximation and interpolation methods designed for multilayered functions. This work presents both a mathematical approach to the functions of that type and their approximation properties.

The second section introduces problems of interpolation and approximation and shows their solutions by RBFs. A multilayered network with RBF units is presented in section 3 together with a description of its learning process with various extensions. Section 4 deals with the regularization method and the Gaussian bar network is described in section 5. Finally, approximation properties of the presented networks are compared with those of multi-layer perceptron-type network.

2. Multidimensional Interpolation and Approximation by RBF

The problem of exact real multivariable interpolation can be formulated as follows:

Given a set of m distinct vectors $\{\mathbf{x}_i; i = 1, \ldots, m\}$ in $\Re^n$ and m real numbers $\{y_i; i = 1, \ldots, m\}$, find a function f satisfying the conditions:

$$f(\mathbf{x}_i) = y_i \quad \forall i = 1, \ldots, m \tag{1}$$

Various interpolation methods use different additional conditions for the family of functions f. The RBF approach introduces the set of m *radial basis functions* ϕ_i,

[1] Kateřina Hlaváčková, Roman Neruda
Institute of Computer Science, Academy of Sciences of the Czech Republic,
P.O.Box 5, 182 07 Prague 8, Czech Republic

$\phi_i : \Re^n \to \Re\ i = 1, \ldots, m$ and uses the form $\phi_i = \phi(\| \mathbf{x} - \mathbf{c}_i \|)$, where $\phi : \Re^+ \to \Re$, $\mathbf{x}\epsilon\Re^n$, $\mathbf{c}_i\epsilon\Re^n$ are the centers of these functions and $\| \cdot \|$ denotes a norm on $\Re^n$ (usually Euclidean). Functions $\phi_i, i = 1, \ldots, m$ form a basis of a linear function space, so the interpolating function f is considered to be their linear combination:

$$f(\mathbf{x}) = \sum_{j=1}^{m} \lambda_j \phi(\| \mathbf{x} - \mathbf{c}_j \|) \tag{2}$$

Substituting the form (2) into the conditions (1), we obtain the following set of linear equations for unknown coefficients λ_j:

$$\begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} \phi_{11} & \cdots & \phi_{1m} \\ \vdots & \ddots & \vdots \\ \phi_{m1} & \cdots & \phi_{mm} \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_m \end{pmatrix} \tag{3}$$

or

$$\mathbf{y} = \mathbf{\Phi}\lambda,$$

where

$$\mathbf{\Phi} = (\phi_{ij})_{i,j} = \phi(\| \mathbf{x}_i - \mathbf{c}_j \|), \quad i, j = 1, \ldots, m.$$

In the case that the matrix $\mathbf{\Phi}$ is regular, we can find the unique solution:

$$\lambda = \mathbf{\Phi}^{-1}\mathbf{y}. \tag{4}$$

The matrix $\mathbf{\Phi}$ can of course be singular for general function ϕ. However, according to results of Micchelli and Powell ([15],[22]), the large class of functions, including the following ones, assures the regularity of the matrix $\mathbf{\Phi}$ (if the data points $\mathbf{x}_i$ are distinct):

1. $\phi(x) = x$;
2. $\phi(x) = x^2 \log x$;
3. $\phi(x) = (x^2 + c)^\alpha, c \geq 0, 0 < \alpha < 1$;
4. $\phi(x) = e^{(-\frac{r}{c})^2}, c \geq 0$;

RBF interpolation scheme can be easily extended to a mapping $F : \Re^n \to \Re^p$. F then equals to $(f_1, \ldots, f_p)$, $\mathbf{y}$ is $(y_1, \ldots, y_p)$ and the condition (1) is generalized to:

$$f_k(x_i) = y_{ik} \quad \forall i = 1, \ldots, m \quad \forall k = 1, \ldots, p. \tag{5}$$

Thus, f_k has the form:

$$f_k(\mathbf{x}) = \sum_{j=1}^{m} \lambda_{jk} \phi(\| \mathbf{x} - \mathbf{c}_j \|) \quad \mathbf{x}\epsilon\Re^n, k = 1, \ldots, p \tag{6}$$

The coefficients λ_{jk} of the linear combination are obtained in the same way as in (4).

The interpolation scheme just defined has the disadvantage that the number of RBFs ϕ_i equals to the number of data points. Typically, one would expect the number of data points to be significantly bigger (approximation problem). Consider that we have m data points and n_0 functions ϕ such that $n_0 < m$. In this case the first problem is how to determine positions of the centers $\mathbf{c}_i, i = 1, \ldots, n_0$. They can either be assigned to some of the data points or be defined in another way. We discuss this problem in section 3 of this contribution.

In this case, when $n_0 < m$, the problem becomes overspecified, the matrix $\mathbf{\Phi}$ is no longer square and a unique inverse does not exist. The solution is found in terms of linear optimization using any of various linear least square methods. Broomhead and Lowe in [3] use the Moore-Penrose pseudo-inverse matrix $\mathbf{\Phi}^+$, which has the property that $\mathbf{\Phi}^+\mathbf{\Phi} = \mathbf{E}_{n_0}$, where $\mathbf{E}_{n_0}$ is the $n_0 \times n_0$ identity matrix and rank $\mathbf{\Phi} = n_0$. Moreover, the solution λ obtained by $\lambda = \mathbf{\Phi}^+\mathbf{y}$ has the smallest norm $\| \lambda \|$ among all the vectors β that minimize the squares sum $\| \mathbf{\Phi}^+\beta - \mathbf{y} \|^2$. The pseudoinverse matrix is computed as:

$$\mathbf{\Phi}^+ = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T. \tag{7}$$

3. Feedforward Network with RBF units

The RBF interpolation and approximation scheme is naturally represented as a three-layer feedforward network with fully interconnected layers. The input layer consists of n units, corresponding to n-dimensional input vectors. Each connection between i-th input node and j-th hidden unit is assigned a weight c_{ji}, which is the i-th coordinate of the j-th unit centre. The hidden unit (also called RBF unit) first computes the distance between the input $\mathbf{x}$ and the center $\mathbf{c}_j$ and then applies the radial function ϕ to this value. The hidden unit's output is connected to the output layer via the weight λ_{jk}. Output units are linear, i.e. they compute weighted sums of their inputs.

A training set for RBF networks consists of vector pairs $(\mathbf{x}_k, \mathbf{y}_k), k = 1, \ldots, m$, where $\mathbf{x}_k \epsilon \Re^n$ presents the input and $\mathbf{y}_k \epsilon \Re^p$ is the desired network output.

Unlike their back-propagation counterparts, RBF networks are trained in a three step process. The first step consists of determining hidden unit centers represented as weights between the input and hidden layer. This problem can be solved easily by choosing n_0 input data randomly and assigning the centers' values to them. Another trivial approach generates the uniform distribution of centers over the input space.

It is required, however, that the centers map the structure of input patterns, so more sophisticated algorithms should be used. The task falls into the unsupervised learning cathegory, which enables us to use various algorithms from clustering to Kohonen feature maps and their variations ([11],[12],[8]). They all have the advantage that they do not use the values $\mathbf{y}_k$ from the training set.

Moody, Darken [16] and others have used the adaptive formulation of k-means clustering algorithm, which looks for a (local) minimum of the overall sum of distances between the centers ('clusters') $\mathbf{c}_j$ and data points $\mathbf{x}_i$:

$$E_1(\mathbf{c}_1, \ldots, \mathbf{c}_{n_0}) = \sum_{i=1}^{m} \sum_{j=1}^{n_0} \delta_{ij} \parallel \mathbf{x}_i - \mathbf{c}_j \parallel^2 \tag{8}$$

δ_{ij} is the cluster membership function represented as a $m \times n_0$ matrix of 0's and 1's with exactly one 1 in each row. $\delta_{ij} = 1$ iff $\mathbf{x}_i$ is a member of the cluster around center $\mathbf{c}_j$. The algorithm itself has the following form:

(i) distribute centers $\mathbf{c}_j$ randomly over the input space

(ii) in time t do the following:

(a) find the center $\mathbf{c}_c$ which is closest to the input $\mathbf{x}_t$

(b) shift the center $\mathbf{c}_c$ towards $\mathbf{x}_t$ according to this formula:

$$\mathbf{c}_c := \mathbf{c}_c + \theta(t) \parallel \mathbf{x}_t - \mathbf{c}_c \parallel, \tag{9}$$

where $\theta(t)$ is a learning rate, having real values between 0 and 1. It is usually set large initially and decreases to zero in time.

An alternative way of determining the centers has been presented by Chen, Cowan and Grant [4] who use the orthogonal least square method for selecting n_0 data points, which minimizes the interpolation error. Coordinates of these data points are then assigned to the centers.

A second optional learning phase sets the additional RBF parameters, if there are any. Let us focus on the very often used Gaussian radial function in the form

$$\phi(x) = e^{-\left(\frac{\|x-c\|}{\sigma}\right)^2} \tag{10}$$

Its parameter σ represents the width of ϕ and thus controls the radial area around the center c in which a certain hidden unit has a reasonable response. The widths effect the generalization capabilities of the network - the smaller they are (comparing with the input set diameter), the worse is the generalization one can expect.

A general way of finding widths' values is by minimizing the error function of the form:

$$E_2(\sigma_1, \ldots, \sigma_{n_0}) = \frac{1}{2} \sum_{r=1}^{n_0} \left[\sum_{s=1}^{n_0} e^{-\left(\frac{\|c_s - c_r\|}{\sigma_r}\right)^2} \left(\frac{\| c_s - c_r \|}{\sigma_r} \right)^2 - P \right]^2 \tag{11}$$

with respect to σ_r (see [17]). The parameter P controls overlapping between areas controlled by different units. In practice, however, various heuristics are used to avoid another minimalization. One of them sets the unit width to the root mean square value of unit distances to its q nearest neighbour units. The number of nearest neighbours is often chosen to be 1. A most simple but frequently used heuristics considers the uniform width of all RBF units. It is important that even the form (11) directly depends neither on the input vectors $\mathbf{y}_i$, nor on $\mathbf{x}_i$. It depends only on the positions of centers $\mathbf{c}_i$, so it is not necessary to present input patterns during the second phase of learning.

After setting weights c_i and RBF unit parameters σ_j, it remains to set the weights λ_{sr}. This is done by the minimalization of the error function:

$$E_3(\Lambda) = \frac{1}{2}\sum_{l=1}^{m} \| y_l - f(x_l) \|^2 == \frac{1}{2}\sum_{l=1}^{m}\sum_{k=1}^{p}(y_{kl} - f_k(x_l))^2 \qquad (12)$$

where $f(\mathbf{x}_l)$ (resp. $f_k(\mathbf{x}_l)$) denotes the network (resp. the k-th output unit) response on the input $\mathbf{x}_l, \Lambda = (\lambda)_{sr}, s = 1, \ldots, p$ and $r = 1, \ldots, n_0$.

By setting the first partial derivatives of E_3 according to λ_{sr} to zero, we obtain the following formula:

$$\lambda_{sr} = \sum_{j=1}^{n_0}(\Phi^+)_{jr}\left[\sum_{l=1}^{m}\phi_j(x_l)y_{kl}\right], \qquad (13)$$

where

$$(\Phi)_{jr} = \sum_{l=1}^{m}\phi_j(x_l)\phi_r(x_l) \qquad (14)$$

and Φ^+ is the Moore-Penrose pseudoinverse (see section 2).

4. Regularization

To improve the generalization capabilities of the network, it is necessary for the mapping realized by network to be sufficiently smooth, which may be a problem especially when the data are noisy. A classical and general approach to controlling the smoothness of the approximating mapping was developed in regularization theory, which can be applied in the third step of the learning process.

An additional term E_R is added to the error function E_3, which now has the form:

$$E_3'(\Lambda) = E_3 + \gamma E_R \qquad (15)$$

E_R is chosen to be large for functions with undesired properties such as big oscillations. This term reflects our implicit assumptions about the desired behaviour of approximating mapping. The real parameter γ controls the ratio between E_3 and E_R. The proper setting of γ is important but task dependent. In general, we can say that by setting γ very small, the term E_R will have no effect, while a large γ will cause the mapping to be very smooth but not to represent the data well.

The concrete form of the regularization term is:

$$E_R(\Lambda) = \frac{1}{2}\sum_{l=1}^{m}\sum_{k=1}^{p}\sum_{i=1}^{n}\left(\frac{\partial^2 f_k(x_l)}{\partial x_i^2}\right)^2; \qquad (16)$$

E_R penalizes functions with large second derivatives and moreover, it is bilinear in the weights λ_{sr}, which retains the advantage of the linear learning algorithm.

By setting the first derivatives of (15) to zero, we can compute the formulas for weights λ_{sr} as in (13). The result is:

$$\lambda_{sr} = \sum_{j=1} n_0(\Psi^+)_{jr} \left[\sum_{l=1}^{m} \phi_j(x_l) y_{kl} \right]; \tag{17}$$

where Ψ has the form of:

$$(\Psi)_{jr} = \sum_{l=1}^{m} \left[\phi_j(x_l)\phi_r(x_l) + \gamma \sum_{i=1}^{n} \frac{\partial^2 \phi_j(x_l)}{\partial x_i^2} \frac{\partial^2 \phi_r(x_l)}{\partial x_i^2} \right] \tag{18}$$

The regularization technique was used by Bishop [20] on the task of recovering the sine function from noisy data. He used an RBF network where the number of hidden units was equal to the number of data points with Gaussian RBFs of uniform weights.

Poggio and Girosi [20] proceeding from general regularization scheme, have derived the so called HyperBF networks with a general Green function used as the activation function ϕ in the hidden units. HyperBF networks include RBF networks as a special case.

5. Gaussian Bars

Each RBF unit responds to a localized region in the input space. On the other hand, perceptrons with a sigmoidal activation function represent the global part made by dividing the input space by more or less fuzzy hyperplane. As a compromise between these two architectures, Hartman and Keeler [7] introduced semi-local units called Gaussian bars.

To describe Gaussian bars, let us first look at the Gaussian RBF from another point of view. A Gaussian function of n-dimensional argument x can be understood as a multiplication of n one-dimensional Gaussian functions. A one-dimensional Gaussian function represents a locality condition for its dimension. The Gaussian RBF unit computes the logical conjuction of these conditions, regarding them to hold simultaneously.

The Gaussian bar unit does not multiply the responses of each input dimension, it computes their weighted sum. This approach corresponds to a logical disjunction of conditions from single dimensions. Thus, one Gaussian bar unit computes the function $\psi : \Re^n \to \Re$

$$\psi(x_1, \ldots, x_n) = \sum_{r=1}^{n} w_r e^{-(\frac{x_r - c_r}{\sigma_r})^2} \tag{19}$$

Gaussian bars were designed in order to avoid slow learning of perceptron type networks and rather non-effective dealing with the irrelevant input dimensions of RBF networks and it shows good performance in some practical tasks. However, in terms of approximation cappabilities, this architecture is weaker than both RBFs and multilayer perceptrons, as was pointed out by Kůrková [14] (see next section).

6. Approximatio

From the theoretica
imation capabilities of
which we can use for d
imation and the capab

An approximation
a set A of approximati
any given function of a

Poggio and Girosi
the best approximatio
functions. (It can, how

Generally, the clas
property. Supposing
the class of functions
approximation propert
centers and widths wh

The universal appr
ical topology by a den
and ρ a metrics on U.
if it is dense in U wi
approximator with re
universal approximato
perceptron with sigmo
([9]) and Kůrková ([13
any continuous multi
sult is also proved by
activation function th
number of units, is a

The universal app
authors ([21],[19],[6]),
radial function ϕ. Mo
networks have the sa

As opposed to mu
works in the form de
ering a more complic
second hidden layer li
Gaussian bar output

References

[1] Bishop C.M.: Imp
works. Neural Co

[2] Braess D.: Nonline

6. Approximation Capabilities

From the theoretical point of view, the important question is to study approximation capabilities of a given network architecture. There are two approaches which we can use for describing those properties: the existence of the best approximation and the capability of universal approximation.

An approximation scheme is said to have the *best approximation property* if in a set A of approximating functions there is one that has minimum distance from any given function of a larger set U.

Poggio and Girosi [21] proved that a multilayered perceptron does not have the best approximation property in the class of real multidimensional continous functions. (It can, however, approximate them arbitrarily well.)

Generally, the class of RBF networks does not have the best approximation property. Supposing the RBF ϕ is Gaussian and has fixed centers and widths, the class of functions realized by the corresponding network then posses the best approximation property. The first two steps of the learning process set values of centers and widths which are fixed in the third step.

The universal approximation property can be described in terms of mathematical topology by a dense subset of a set. Let U be a class of functions, T its subset and ρ a metrics on U. The class T is called an *approximator* with respect to (U, ρ) if it is dense in U with respect to the topology induced by ρ. If class T is an approximator with respect to the class of continous real functions, it is called a *universal approximator*. The question of universal approximation of a multilayered perceptron with sigmoidal nonlinearities was investigated by Cybenko ([5]), Hornik ([9]) and Kůrková ([13]), who found that functions of that type can approximate any continuous multidimensional function arbitrarily well ([9]). The strongest result is also proved by Hornik, who claims that for any bounded and non-constant activation function the set of networks with one hidden layer, having an arbitrary number of units, is a universal approximator [9].

The universal approximation property for RBF has been proved by several authors ([21],[19],[6]), whose claims slightly differ in their assumptions about the radial function ϕ. Moreover, Poggio and Girosi ([21]) have shown that HyperBF networks have the same property.

As opposed to multilayered perceptron and RBF networks, Gaussian bar networks in the form described above are not universal approximators [14]. Considering a more complicated network architecture, one Gaussian bar hidden layer, a second hidden layer linear (computing only weighted sums of their inputs) and a Gaussian bar output layer, the universal approximation property is preserved.

References

[1] Bishop C.M.: Improving the generalization properties of radial basis function neural networks. Neural Computation 3, 1991, 579-588.

[2] Braess D.: Nonlinear approximation theory. Springer, Berlin, Heidelberg, New York, 1986.

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

[4] Chen S., Cowan C.F.N., Grant P.M.: Orthogonal least squares learning algorithm for radial basis function networks. IEEE Transactions on Neural Networks, **2**, 2, 1991.

[5] Cybenko G.: Approximation by superposition of a sigmoidal function. Mathematics of Control, Signals and Systems, 2, 1989, 303-314.

[6] Hartman E.J., Keeler J. D., Kowalski J. M.: Layered neural networks with Gaussian hidden units as universal approximations. Neural Computation 2, 1990, 210-215.

[7] Hartman E. J., Keeler J. D.: Predicting the future: advantages of semilocal units. Neural Computation 3, 1991, 566-578.

[8] Hlaváčková K.: On some variants of adaptive rules of feature maps. Neural Network World, 5, IDG, 1991, 287-293.

[9] Hornik K.: Approximation capabalities of multilayer feedforward networks. Neural Networks, 2, 1991, 251-257.

[10] Kittler J., Devijver P.A.: Pattern recognition: A statistical approach. Prentice-Hall International, Inc., London, 1982.

[11] Kohonen T., Ritter H.: Self-organizing semantic maps. Biol. Cybern.61, 1989, 241-254.

[12] Kohonen T.: The self-organizing map. Proc. of the IEEE, **78**, 9, September 1990.

[13] Kůrková V.: Kolmogorov's theorem and multilayer neural networks. Neural Networks, 5, 1992, 501-506.

[14] Kůrková V.: Universal approximation using feedforward neural networks with Gaussian bar units. Proceedings of ECAI '92, Vienna, 1992, 193-197.

[15] Micchelli C. A.: Interpolation of scattered data: Distance matrices and conditionally positive definite functions. Construct. Approx. 2, 11, 1986.

[16] Moody J., Darken C.: Learning with localized receptive fields. In: Proceedings of the 1988 Connectionist Models Summer School, D. Touretzky, G. Hinton, and T. Sejnowski, eds. Morgan Kaufmann, San Mateo, CA, 1989a.

[17] Moody J., Darken Ch. J.: Fast adaptive k-means clustering: some empirical results. Proc. IJCNN, San Diego '90, **2**, 1990, 233-238.

[18] Nash J.C.: Compact numerical methods for computers: linear algebra and function minimization. Adam-Hilger Ltd., Bristol, 1980.

[19] Park J., Sandberg I. W.: Universal approximation using radial-basis-function networks. Neural Computation 3, 1991, 246-257.

[20] Poggio T., Girosi F.: Networks for approximation and learning. Proceedings of the IEEE, **78**, 9, September 1990.

[21] Poggio T., Girosi F.: Networks and the best approximation property. Biological Cybernetics, **63**, 1990, 169-176.

[22] Powell M. J. D.: The theory of radial basis function approximation in 1990. DAMPT/1990/NA11, 1990.

[23] Rumelhart D.E., Hinton G.E., Williams R.J.: Learning internal representations by error propagation. Parallel Distributed Processing, ch. 8, MIT Press, Cambridge, MA, 1986, 318-362.

[24] Stinchcombe M., Wh
sigmoid hidden layer
on Neural Networks.
1/607-611.

[25] Stone C. J.: Optimal
10, 1982, 1040-1053.

[24] Stinchcombe M., White H.: Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. Proceedings of the Intern. Joint. Conference on Neural Networks. Washington DC, IEEE TAB Neural Network Commitee, June 1989, 1/607-611.

[25] Stone C. J.: Optimal global rates of convergence for non-parametric regression. Ann. Stat., **10**, 1982, 1040-1053.