A COMPLETE SEQUENTIAL LEARNING ALGORITHM FOR RBF NEURAL NETWORKS with APPLICATIONS

NICOLAE POPOVICIU
Hyperion University of Bucharest, Mathematics-Informatics Faculty
021 156 Bucharest, Sos. Colentina 25A, Apt 7, ROMANIA
MIOARA BONCUȚ
University Lucian Blaga of Sibiu
Sibiu, Str. Dealului 29, ROMANIA

*Abstract***.** A multidimensional approximation problem and the method to solve it is the theme of this work. The problem solving is done by a RBF neural network and, for that, a RBF Sequential Learning Algorithm is described, based on [4]. Section 1 gives the mathematical background and the notations used in the algorithm description. The initialization of this algorithm is done by another algorithm, called the c-means algorithm, from section 2. The sections 3 and 4 contain the RBF sequential learning algorithm itself, in comparison with the RBF batch learning algorithm [4], [7]. A change in the initialization of weight vectors is done, versus [4]. The section 5 contains a numerical example, with all steps of c-means algorithm and several steps of RBF sequential learning algorithm.

*Key-Words***.** Radial basis function, RBF, sequential learning algorithm, RBF neural network, gradient descent, c-means initialization algorithm.

1. Mathematical Background and Notations

Having the input data (input vectors, training set, learning set)

$$I(x,y)=\{x(t)\in R^n, y(t)\in R^M \ / t=1,N\}$$

$$x(t)=(x_1(t)=x_{1t}\cdots\cdots x_n(t)=x_{nt})^T$$

$$y(t)=(y_1(t)=y_{1t}\cdots\cdots y_M(t)=y_{Mt})^T$$

we look for the function $s: R^n \to R^M$ so that

$s[x(t)]=y(t)$, $t=1,N$.

In order to obtain the best fit for this approximation model the work [4], page 6, proposes a RBF neural network with one hidden layer Sh and the following linear interpolation model

$$w_{0k}+\sum_{j=1}^{c} g(\|x(t)-v(j)\|^2)w_{jk}=y_k(t) \qquad (1)$$

where: k runs from 1 to M ; namely $k=1,M$;

M is the number of output units in the output layer

So : $x(t),v(j)\in R^n$ for $t=1,N; j=1,c$;

$g: R_+ \to R$, $g=g(\alpha)$ is a **radial basis function (RBF)** [4], [1], [6]**;**

c is the number of radial basis functions in the hidden layer Sh; w_{jk} are the **unknown** weights;

w_{0k} is the output offset (also unknown);

$v(j)$ $j=1,c$ are **the prototypes** [4] of the input vectors $x(t)$, called also **the centers** [1], page 73, associated with the radial basis functions of hidden layer Sh. Hence $v(1),\cdots,v(j),\cdots,v(c)$ are the **locations** ([4], page 31) or the **positions** ([4], page 2) of the radial basis functions $g\left(\|x(t)-v(j)\|^2\right)$.

The linearity in (1) is understood in relation with the unknowns w_{jk} .

The main task of the interpolation problem (1) is to find the unknowns $w_{jk}, j=0,c; k=1,M$. In order to do this, the RBF neural network is trained using learning algorithms based on **gradient descent** [4], as we use in this work.

The model (1) enhances some main questions: how can we choose the natural number c , the function g and how can be initialized the RBF centers $v(j), j=1,c$? For c we suggest that $2c\le N$. The function g is a radial basis function [4], [1], [6]

$$g(\alpha)=(g_0(\alpha))^{1/(1-m)} \ , \ m>1, g_0(\alpha)>0 \ .$$

In any concrete problem the number c and the function g are chosen only one time. They remain unchanged during the learning process, while the RBF centers are updated step by step. That is why the centers must begin by initialization.

For the center $v(j)$, $j=1,c$ of the j th unit from hidden layer Sh there are several initialization possibilities [4] , page 31:

a). random generating; b). using a set of prototypes specifically determined to represent the input vectors $x(t)$ used in the training set;

c). using the **c-means algorithm** [4], page 32.

2. The c-Means Initialization Algorithm

The c-means initialization algorithm (or shortly, c-means algorithm) begins with an arbitrary initial set of c prototypes $v_0(1),\cdots,v_0(j),\cdots,v_0(c)$ from R^n, which implies the partition of the input vectors $x(t)$ into c clusters. Each cluster is represented by a prototype and each input vector is assigned to a cluster whose prototype is its closest neighbor. Hence, having the initial set of prototypes we have to construct the centers $v(1),\cdots,v(j),\cdots,v(c)$. In order to do that, one defines the **indicator function**

$u_j : \{x(t), t=1,N\} \to \{0,1\}$, $j=1,c$

$u_j[x(t)] = u_{tj}$.

The values of the indicator function are computed during several steps. We indicate that by a DO loop programming statement, as it follows

$DO \;\; a \;\; t=1,N$

$$x(t)-v_0(j_0) = \min_{j=1,c}\left\{\left\| x(t)-v_0(j)\right\|^2\right\}$$

$u_{t\,j_0} = 1$

$u_{tj} = 0$, $j=1,c; j\neq j_0$; $1\le j_0\le c$

a CONTINUE.

Here a is an arbitrary label to indicate the place where t becomes $t+1$. We can define the matrix $U=(u_{tj})$, $U=U_{Nxc}$, $j=1,c$. The cluster j contains all input vectors $x(t)$ having $u_{tj}=1$.

Having the calculated values u_{tj} of the indicator function, we calculate the set of RBF centers $v(j)$, $j=1,c$ as it follows

$$v(j) = \left(\sum_{t=1}^{N} x(t)u_{tj}\right) / \left(\sum_{t=1}^{N} u_{tj}\right) \qquad (2)$$

The denominator of (2) is a non-zero value, because each input vector $x(t)$ belongs to one cluster. When $x(t)$ may belong to many clusters, then we chose the class with the smallest index.

We summarize the steps of c-means algorithm.

Step a. We arbitrarily choose the initializing vectors $v_0(1),\cdots,v_0(j),\cdots,v_0(c)$.

Step b. Compute the values of the indicator function by DO loop statement. **Step c.** Compute the vectors (centers) $v(j)$, $j=1,c$ by (2).

3. Learning Algorithms Based on Gradient Descent. Notations

There are two RBF learning algorithms based on gradient descent.

1). Batch learning algorithm [4], presented with examples in [7]. 2). Sequential learning algorithm [4].

Both algorithms use the same matrices for a **supervised learning process**. First we supply the matrix notations and all dimensions and then we describe the meaning of matrix elements.

Generally, for a matrix A we use the notations : $A=(a_{ij}), i=1,m; j=1,n$; $A=A_{mxn}$ with m lines (rows) and n columns ; $a_j \in R^m$ is the j th column vector of A ; $la_i \in R^n$ is the i th line of A ; A^T , T means transpose . Always a vector is a column vector.

Specifically, related with our theme, we use the following matrices.

$$g_{tj} = g(\left\| x(t)-v(j)\right\|^2), t=1,N; j=1,c$$
$$g_{t0} = 1, t=1,N \qquad (3)$$
$$G=(g_{tj}), G=G_{Nx(c+1)}$$

Related with the run of index j we notice that sometimes $j=0,c$ and sometimes $j=1,c$.

$$G = \begin{pmatrix} 1 & g_{11} \cdots g_{1c} \\ \cdots\cdots\cdots\cdots\cdots \\ 1 & g_{N1} \cdots g_{Nc} \end{pmatrix} .$$

Also we use the matrices :

$X = (x_i(t) = x_{it}), X = X_{nxN}$;

$Y = (y_k(t) = y_{kt}), Y = Y_{MxN}, Y^T = Y^T_{NxM}$

$\hat{Y} = (\hat{y}_{kt}), \hat{Y} = \hat{Y}_{MxN}$;

$V = (v_i(j) = v_{ij}), V = V_{nxc}$;

$W = (w_{jk}), W = W_{(c+1)xM}, j = 0, c$;

$P = (p_{tk}), P = P_{NxM}$;

$Q = (q_{tj}), Q = Q_{Nxc}$.

The meanings of matrices elements will be clarified in the future.

The linear **RBF model** (1) becomes

$$w_{0k} + \sum_{j=1}^{c} g_{tj} w_{jk} = y_{kt} \qquad (4)$$

We pay attention that in (4) the indexes t and k changed their positions by passing in the right hand side.

The batch learning algorithm is developed by using the gradient descent to minimize the total error

$$E = \frac{1}{2} \sum_{k=1}^{M} \sum_{t=1}^{N} (y_{kt} - \hat{y}_{kt})^2 .$$

The sequential learning algorithm is developed by using the gradient descent to minimize partial errors

$$E_t = \frac{1}{2} \sum_{k=1}^{M} (y_{kt} - \hat{y}_{kt})^2 \ , \ t = 1, N .$$

Remark 1. These basic differences generate different formulas to update the vectors w_k and $v(j)$ in the corresponding algorithms.

IV. RBF Sequential Learning Algorithm

Our aim is to obtain the mapping

$R^n \to R^M$, $x(t) \to y(t)$, $t = 1, N$.

This is done by RBF linear model (4). For some given weights w_{0k} , w_{jk} and centers (prototypes) $v(j), j = 1, c$ in equations (4) we have only the approximation

$$w_{0k} + \sum_{j=1}^{c} w_{jk} g(\| x(t) - v(j) \|^2) \approx y_{kt}$$

which is denoted

$$w_{0k} + \sum_{j=1}^{c} w_{jk} g(\| x(t) - v(j) \|^2) = \hat{y}_{kt} \qquad (5)$$

The RBF network training is typically based on the minimization of the **total error** between the given **desired responses** $y_k(t) = y_{kt}$ and the **actual responses** $\hat{y}_{kt}$ from (5). For any two vectors u and v belonging to the same vector space, let us say $u, v \in R^n$, we denote the scalar product by

$$< u, v >= u \cdot v = u^T v = \sum_{i=1}^{n} u_i v_i \ .$$

In this work, the describing of Sequential Learning algorithm (SL algorithm or RBF SL algorithm) will be based mainly on vector and matrix writing, rather than component form, as some works do. Nevertheless, we mention the meanings of all matrix components. So we mention that

g_{tj} represents **the response of the j th RBF** associated in hidden layer with the j th unit (center) to the input vector $x(t)$;

$\hat{y}_{kt}$ represents the **actual response of the RBF network** associated in output layer with the k th output unit to the same input vector $x(t)$. The present RBF SL algorithm is based on [4], 6.2, but we changed the initialization step for the weights w_k .

The RBF SL Algorithm

Step 1. Introduce the input data

a). Scalar data: $n, N; M; c; \varepsilon, \eta$ (learning rate);

Hence, the indexes variation is

$i = 1, n; t = 1, N$; $k = 1, M; j = (0)1, c$.

b). Vector data $I(x, y)$. Construct the matrices

$X = (x_i(t) = x_{it}), X = X_{nxN}$;

$Y = (y_k(t) = y_{kt}), Y = Y_{MxN}, Y^T = Y^T_{NxM}$.

c). An arbitrary initial set of c prototypes

$v_0(1), \cdots, v_0(j), \cdots, v_0(c)$ from R^n .

Step 2. a). Choose the RBF $g(\alpha)$.

b). Initialize all the RBF centers (prototypes) $v(j), j = 1, c$ by using the c-means algorithm.

Step 3. a). Compute the vectors

$x(t) - v(j), t = 1, N; j = 1, c$.

b). Compute all the norms $\| x(t) - v(j) \|^2$.

c). Compute $g_{t\,j} = g\left(\| x(t) - v(j) \|^2 \right)$,

$g_{t\,0} = 1$; $G = (g_{t\,j})$, $G = G_{Nx(c+1)}$.

Step 4. Initialize the weights vectors $w_k \in R^{c+1}$.

Version 1. Karayiannis - Behnke [4] set

$w_k = \theta \in R^{c+1}$, θ null vector.

Version 2. We solve separately (for example, by Mathcad) M linear systems of the form

$$Gw_k = ly_k^T \ , \ k = 1, M \qquad (6)$$

$w_k = (w_{0k}\ w_{1k} \cdots w_{jk} \cdots w_{ck})^T$

$ly_k = (y_{k1}\ \ y_{k2} \cdots y_{kt} \cdots y_{kN})$.

Remark 2. Others equivalent forms of (6) are

$w_{0\,k} + \sum_{j=1}^{c} g_{t\,j} w_{j\,k} = y_{k\,t}$ or

$w_{0\,k} + < l\,g_t, w_{k>} = y_{k\,t}$, $t = 1, N; k = 1, M$.

The solving of system (6) is done by the least squares method, which leads to the normal Gauss solution having the form

$$w_k = \left(G^T G \right)^{-1} G^T ly_k^T \qquad (7)$$

We denote this least square solution by $w_k = \hat{w}_k$

and the matrix W is (on columns)

$W = (w_1 \cdots w_k \cdots w_M)$ or (on lines) lw_k .

Step 5. Compute the actual RBF neural network responses, denoted $\hat{y}_{k\,t}$.

Version 1. We obtain $\hat{y}_{k\,t} = 0$, $\hat{Y} = O$.

Version 2. We obtain

$$l\hat{y}_k^T = Gw_k \text{ or } \hat{y}_{k\,t} = < l\,g_t, w_k > \qquad (8)$$

and construct the matrix $\hat{Y} = (\hat{y}_{k\,t})$, $\hat{Y} \neq O$.

Step 6. Compute the output errors $p_{k\,t}$ of the RBF neural network and construct the matrix P .

Version 1. $P = Y^T - O^T = Y^T$.

Version 2. $P = Y^T - \hat{Y}^T$ or $p_{t\,k} = y_{k\,t} - \hat{y}_{k\,t}$

$P = (p_1 \cdots p_k \cdots p_M)$, $p_k \in R^N$.

Remark 3. In version 2 we have

$\sum_{t=1}^{N} p_{t\,k} = 0$, $k = 1, M$

due to the least square method.

Step 7. Compute the total error E

$E = \frac{1}{2} \sum_{t=1}^{N} \sum_{k=1}^{M} (p_{t\,k})^2$ or

$$E = \frac{1}{2} \left(p_1^T p_1 + \cdots + p_k^T p_k + \cdots p_M^T p_M \right) \qquad (9)$$

Step 8. If $E < \varepsilon$ then STOP. Print the weights vectors $w_k, k = 1, M$. Otherwise, continue i. e. GO TO Step 9. **Step 9.** Set $E^{old} = E$.

Step 10. Update the weight vectors $w_k \in R^{c+1}$ and the center vectors $v(j) \in R^n$.

DO *d* $t = 1, N$

$g_{t\,0} = 1$

* Compute the line lg_t from G by using a single $x(t)$ and all $v(j)$.

DO *a* $j = 1, c$

compute $x(t) - v(j)$; $\| x(t) - v(j) \|^2$

$g_{t\,j} = g\left(\| x(t) - v(j) \|^2 \right)$

a CONTINUE

* Update the whole matrix W , i.e. replace the current estimate of each weight vector w_k (w_k old) by its updated version w_k (w_k new).

DO *b* $k = 1, M$

$\hat{y}_{k\,t} = < w_k, l\,g_t >$

$p_{t\,k} = y_{k\,t} - \hat{y}_{k\,t}$ (the RBF network errors)

$w_k = w_k + \eta\, p_{t\,k}\, lg_t^T$

b CONTINUE

* Update all center vectors, i.e. replace the current estimate of each $v(j)$ ($v(j)$ old) by its updated version $v(j)$ (the new $v(j)$) .

DO *c* $j = 1, c$

$$q_{t\,j} = \frac{2}{m-1} g_0' \left(\| x(t) - v(j) \|^2 \right) (g_{t\,j})^m \cdot \cdot < lp_t, lw_j > \qquad (10)$$

(the hidden errors produced by RBF number j for one $x(t)$) ; $v(j) = v(j) + \eta\, q_{t\,j} \left[x(t) - v(j) \right]$

c CONTINUE ; *d* CONTINUE .

Remark 4. The neural network which is trained according to this scheme uses the propagation back the output errors $p_{tk} = y_{kt} - \hat{y}_{kt}$.

Step 11. Compute the new matrix G by using the updated vectors

$w_1, \cdots, w_k, \cdots, w_M$ and $v(1), \cdots, v(j), \cdots, v(c)$.

a). $x(t) - v(j), t = 1, N; j = 1, c$,

b). $\| x(t) - v(j) \|^2$, c). $g_{tj} = g\left(\| x(t) - v(j) \|^2 \right)$

$g_{t0} = 1$, $G = (g_{tj}), G = G_{Nx(c+1)}$.

Step 12. Compute the new actual RBF neural network responses

$\hat{y}_{kt} =< w_k, lg_t >$, $\hat{Y} = \left(\hat{y}_{kt} \right), \hat{Y} = \hat{Y}_{MxN}$.

Step 13. Compute the new RBF neural network errors contained in matrix P

$P = Y^T - \hat{Y}^T, P = \left(p_{tk} \right)$.

Step 14. Compute the new total error

$$E = \frac{1}{2}\left(p_1^T p_1 + \cdots + p_k^T p_k + \cdots p_M^T p_M \right).$$

Step 15. Decision step. If

$\left(E^{old} - E \right) / E^{old} < \varepsilon$ then print the main results w_k , $v(j)$. STOP; otherwise GO TO Step 9.

5. Numerical Example

In this numerical example we apply the above RBF SL algorithm, with a Gaussian RBF [6]

$$g_0(\| x(t) - v(j) \|^2) = e^{\| x(t) - v(j) \|^2} \text{ , } m = 2$$

$$g(\| x(t) - v(j) \|^2) = e^{-\| x(t) - v(j) \|^2} \qquad (11)$$

$$g_{tj} = e^{-\| x(t) - v(j) \|^2} \text{ , } t = 1, N; j = 1, c$$

$$g_{t0} = 1$$

and q_{tj} obtained from (10) with the form

$$q_{tj} = 2 g_{tj} < lp_t, lw_j > \qquad (12)$$

Problem. Find a function which maps

$$R^3 \to R^2, x(t) \to y(t)$$

for the following input data

t	1	2	3	4	5
$x_1(t)$	0	1	1	0	1
$x_2(t)$	0	0	1	1	1
$x_3(t)$	0	1	0	1	1
$y_1(t)$	1	0	−1	1	1
$y_2(t)$	0	1	1	−1	−2

$x(t) = \left(x_1(t) \;\; x_2(t) \;\; x_3(t) \right)^T$

$y(t) = \left(y_1(t) \;\; y_2(t) \right)^T$.

Solution. For this problem we simply apply the whole c-means initialization algorithm –beginning with the case 3 , below - and we do several steps of RBF SL algorithm.

Case 1. Step 1. Input data

$n = 3, N = 5; M = 2; c = 2; \varepsilon = 0.5; \eta = 1$;

$I(x, y)$.

Step 2. Use a Gaussian RBF (11) .

We arbitrarily choose the RBF centers $v(j)$

$v(1) = (-1/2 \;\; 1/2 \;\; -1)^T$, $v(2) = (1 \;\; 2/3 \;\; 0)^T$

Step 3. Compute $x(t) - v(j), \| x(t) - v(j) \|^2$ and we obtain the matrix G

$$G = \begin{pmatrix} 1 & 0.2231 & 0.2359 \\ 1 & 0.0111 & 0.2359 \\ 1 & 0.0302 & 0.8948 \\ 1 & 0.0821 & 0.1211 \\ 1 & 0.0111 & 0.3292 \end{pmatrix}.$$

Step 4. We apply version 2 and solve two linear systems $Gw_1 = ly_1^T$, $Gw_2 = ly_2^T$.

The result is the matrix W

$$W = (w_1 \quad w_2) = \begin{pmatrix} 1.098 & -1.158 \\ 2.0 & 2.534 \\ -2.314 & 2.137 \end{pmatrix}.$$

Step 5. Compute the matrix $\hat{Y}$, version 2.

Step 6. Compute the matrix P , version 2, i.e. output errors .

Step 7. Compute the total error (9) , $E = 3.001$.

Step 8. Continue.

Case 2. The RBF SL algorithm uses the same input

data as before, except the centers $v(j)$. Geometrically, we choose the following RBF centers

$v(1) = (1/2 \;\; 0 \;\; 1/2)^T$, $v(2) = (1/2 \;\; 1 \;\; 1/2)^T$.

We obtain the matrices

$$G = \begin{pmatrix} 1 & 0.6065 & 0.2231 \\ 1 & 0.6065 & 0.2231 \\ 1 & 0.2231 & 0.6065 \\ 1 & 0.2231 & 0.6065 \\ 1 & 0.2231 & 0.6065 \end{pmatrix}, \; W = \begin{pmatrix} 0.367 & -0.471 \\ 0.573 & 1.726 \\ 0.158 & -1.229 \end{pmatrix}$$

$\det G^T G \approx 0$.

The total error is

$E = \frac{1}{2}(3.492 + 5.328) = 4.41$. Continue.

Case 3. Step 1. Input data

a) $n = 3, N = 5; M = 2;\; c = 2;\; \varepsilon = 0.5; \eta = 1$;

b) $I(x, y)$; c) We need an arbitrary set of $c = 2$ prototypes $v_0(1), v_0(2)$.

Step 2. a). Use a Gaussian RBF (11) .

b). Initialize all the RBF centers (prototypes) $v(j), j = 1, c$ by using the **c-means algorithm.**

Step a. We arbitrarily choose the initializing vectors $v_0(1), \cdots, v_0(j), \cdots, v_0(c)$. So we take

$v_0(1) = (1/2 \;\; 0 \;\; 1/2)^T$, $v_0(2) = (1/2 \;\; 1 \;\; 1/2)^T$.

Step b. Compute the values of the indicator function $u_j[x(t)] = u_{t\,j}$ by DO loop statement. So we obtain successively

$x(t) - v_0(1)$ and $\| x(t) - v_0(1) \|^2$, $t = 1, N$

$\| x(t) - v_0(1) \|^2$: 0.5 0.5 1.5 1.5 1.5

$\| x(t) - v_0(2) \|^2$: 1.5 1.5 0.5 0.5 0.5 .

Compute $\min\left(\| x(t) - v_0(1) \|^2 \right)$ for $t = 1,5$

$\min\left(\| x(t) - v_0(2) \|^2 \right)$ for $t = 1,5$

and obtain the values $u_{t\,j}$ of matrix U as it follows

$u_{t\,j}$	$j=1$	$j=2$			
$x(1)$	1	0	$x(4)$	0	1
$x(2)$	1	0	$x(5)$	0	1
$x(3)$	0	1			

Step c. Compute the vectors (centers) $v(j)$, $j = 1, c$ by (2)

$v(1) = \frac{x(1) + x(2)}{2} = (1/2 \quad 0 \quad 1/2)^T$

$v(2) = \frac{x(3) + x(4) + x(5)}{3} = (2/3 \quad 1 \quad 2/3)^T$

Here begins the Step 3 . The total error will be less then in the cases 1 and 2 . The algorithm have to continue in order to improve the error.

6. Conclusions

The algorithm works well on numerical examples. But a successful learning process depends on the appropriate choosing of c, ε, η and the RBF g . That is why the RBF SL algorithm must be implemented in a computer program, which gives the possibility to do a lot of experiments.

References

[1]. BACK Andrew, *Radial Basis Functions*, Chapter 3 in Handbook of Neural Network Signal Processing, Edited by Yu Hen Hu, Jenq-Neng Hwang, CRC Press, 2002.

[2]. FREEMAN A. James, SKAPURA M. David, *Neural Networks*: *Algorithms, Applications and Programming Techniques*, Addison-Wesley Publishing Company, 1991.

[3]. JAIN K. Anil, MAO Jianchang, MOHIUDDIN K. M. , *Artificial Neural Networks: A Tutorial*, IEEE, March, 1996.

[4]. KARAYIANNNIS N. B. , BEHNKE S. , *New Radial Basis Neural Networks and their Applications in a Large-Scale Handwritten Digit Recognition Problem*, Chapter 2, CRC Press LL3, 2000.

[5]. KARAYIANNIS N. B. , *Reformulated Radial Basis Neural Networks Trained by Gradient Descent*, IEEE Transactions on Neural Networks, vol. 10, no. 3, page 657-671, 1999.

[6]. POPOVICIU Nicolae, MOISE M. *Radial Basis Functions. Foundations and Examples. Part I* , CD and Proceedings of XI th RAU Symposium , May 2006, page 68-73, ISBN (10)-973-7854-46-2, Bucharest.

[7]. POPOVICIU Nicolae, *Radial Basis Functions. Foundations and Examples. Part II.* To appear.