PERCEPTRON NETWORK FOR RBF LOVERS

*Josef Bostik, Jaromir Kukal, Miroslav Virius**

Abstract: There are two basic types of artificial neural networks: Multi-Layer Perceptron (MLP) and Radial Basis Function network (RBF). The first type (MLP) consists of one type of neuron, which can be decomposed into a linear and sigmoid part. The second type (RBF) consists of two types of neurons: radial and linear ones. The radial basis function is analyzed and then used for decomposition of RBF network. The resulting Perceptron Radial Basis Function Network (PRBF) consists of two types of neurons: linear and extended sigmoid ones. Any RBF network can be directly converted to a four-layer PRBF network while any MLP network with three layers can be approximated by a five-layer PRBF network. The new PRBF network is then a generalization of MLP and RBF network abilities. Learning strategies are also discussed. The new type of PRBF network and its learning via repeated local optimization is demonstrated on a numerical example together with RBF and MLP for comparison. This paper is organized as follows: Basic properties of MLP and RBF neurons are summarized in the first two chapters. The third chapter includes novel relationship between sigmoidal and radial functions, which is useful for RBF decomposition and generalization. Description of new PRBF network, together with its properties, is subject of the fourth chapter. Numerical experiments with a PRBF and their requests are given in the last chapters.

Key words: *ANN, perceptron, RBF, sigmoidal function, decomposition, optimization*

Received: November 10, 2011
Revised and accepted: November 6, 2012

1. MLP Preliminaries

General *Multi-Layer Perceptron* (MLP) [1] network consists of a single input layer, at least one hidden layer and a single output layer. The signal processing in every hidden and output neuron is described by formula $y = f\left(\sum_{k=0}^{n} w_k x_k\right)$,
where $n \in \mathrm{N}$ is the number of neuron inputs,
$x_k, w_k \in \mathbf{R}$ are k^{th} input value and its weight,
$y \in [a; b]$ is the neuron output, $x_0 = 1$
and $f : \mathbf{R} \to [a; b]$ is a non-decreasing continuous function satisfying:

*Josef Bostik, Jaromir Kukal, Miroslav Virius
FNSPE CTU Prague, Brehova 7, 11519 Praha 1, Phone: 0042 732 722 861, Fax: 0042 224 918 643, E-mail: josefbostik@gmail.com, jaromir.kukal@fjfi.cvut.cz, miroslav.virius@fjfi.cvut.cz

- $f(s) + f(-s) = a + b$
- f is concave on $\mathbf{R}_0^+$
- $\lim_{s \to +\infty} f(s) = b$
- $f''(0)$ exists
- $\lim_{s \to +0} f''(s)$ exists

This non-linear function f is called *sigmoid function*. The other properties of sigmoid function can be easily derived:

- $f(0) = (a + b)/2$
- f is convex on $\mathbf{R}_0^-$
- $\lim_{s \to -\infty} f(s) = a$
- $\lim_{s \to 0-} f''(s) = -\lim_{s \to 0+} f''(s)$

Traditional example of sigmoid function is *logistic function* $f_L(s) = (1 + \exp(-s))^{-1}$, which maps $\mathbf{R}$ into $(0; 1)$ and $f_L(0) = 1/2, f_L'(0) = 1/4$. The sigmoid function will be useful for RBF decomposition when $a = 0$, $b = 1$, $f'(0) = 1$. There are examples of such sigmoid functions:

$$f_1(s) = \frac{1 + \tanh 2s}{2}$$

$$f_2(s) = \frac{1}{2} + \frac{1}{\pi} \arctan\ \pi s$$

$$f_3(s) = \frac{1}{2} + \frac{s}{1 + 2|s|}$$

$$f_4(s) = \min\left(1, \max\left(0, \frac{1}{2} + s\right)\right)$$

$$f_5(s) = \frac{1}{2} + \frac{1}{2} \operatorname{erf} \sqrt{\pi} s,$$

where $\operatorname{erf} x = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$.

2. RBF Preliminaries

General *Radial Basis Function* (RBF) [1, 2] network consists of three layers: input, hidden and output ones. The signal processing in every output neuron is described by linear formula $y = \sum_{k=0}^{n} w_k x_k$, where $n \in \mathbf{N}$ is the number of neuron inputs, $x_k, w_k \in \mathbf{R}$ are k^{th} input value and its weight, $y \in \mathbf{R}$ is the neuron output and $x_0 = 1$. The signal processing in every hidden neuron is described by formula

$y = \exp\left(-\frac{1}{2\sigma^2}\sum_{k=1}^{n}(x_k - w_k)^2\right)$, where $\sigma > 0$ is the space factor, $y \in (0;1]$ is the neuron output and the other quantities have the same meaning. The substitution $s_k = \frac{x_k - w_k}{\sqrt{2}\sigma}$ enables to study the properties of hidden neuron. Thus, $y = G(s_1, \ldots, s_n) = \exp\left(-\sum_{k=1}^{n} s_k^2\right)$, where $G : \mathbf{R}^n \to (0;1]$ is *radial function.* The second advantage of function G is its separability. Let $g_R(s) = \exp\left(-s^2\right)$, then $G(s_1, \ldots, s_n) = \Pi_{k=1}^{n}\ g_R(s_k)$ is separable in Cartesian coordinates. It is necessary to collect generalized properties of function g, which will be useful for multiplicative construction of function G.

Let $g : \mathbf{R} \to [0;1/4]$ be continuous function satisfying:

- $g(s) = g(-s)$
- $g(0) = 1/4$
- g is non-increasing on $\mathbf{R}_0^+$
- $\lim_{s\to+\infty} g(s) = 0$
- $g''(0) = -2$

Such function g can be called *base function.*

The other properties of base function can be easily derived:

- g is non-decreasing on $\mathbf{R}_0^-$
- $\lim_{s\to-\infty} g(s) = 0$

Here are examples of base functions:

$$g_1(s) = \frac{1}{4}\exp\left(-4s^2\right)$$

$$g_2(s) = \frac{1}{4}\left(1 + 4s^2\right)^{-1}$$

$$g_3(s) = \frac{1}{4}\max\left(0, 1 - 4s^2\right).$$

They can form separable functions G_1, G_2, G_3, but G_1 is the only radial symmetric function.

Having our favorite sigmoid function, we can construct base function $g(s)$ as a product of $f(s)$ and $f(-s)$ but unfortunately, the radial property is not guaranteed.

Theorem 1: Let f be sigmoid function with $a = 0$, $b = 1$, $f'(0) = 1$. Then the function $g(s) = f(s)\, f(-s)$ is base function.

Base function can be used for the approximation of RBF neuron using formula $G(s_1, \ldots, s_n) = \Pi_{k=1}^{n} f(s_k)\, f(-s_k)$.

3. New Perceptron Formula for RBF Design

Let $g(s) = \frac{1}{4}\exp\left(-4s^2\right)$ be known and $f(s)$ be unknown sigmoid function with

$$a = 0\,, b = 1\,, f'(0) = 1.$$

Then

$$f(s) f(-s) = \frac{1}{4}\exp\left(-4s^2\right)$$

$$f(s)\,(1 - f(s)) = \frac{1}{4}\exp\left(-4s^2\right)$$

$$f^2(s) - f(s) + \frac{1}{4}\exp\left(-4s^2\right) = 0.$$

The only acceptable solution of given quadratic equation is $f(s) = \frac{1+\mathrm{sign}(s)\sqrt{1-\exp(-4s^2)}}{2}$, which is a new kind of sigmoid function.

The existence of new sigmoid function $f_6(s) = \frac{1+\mathrm{sign}(s)\sqrt{1-\exp(-4s^2)}}{2}$ is in contradiction to traditional logistic function $f_1(s) = \frac{1+\tanh 2s}{2}$. The absolute difference between them can be denoted as $D(s) = |f_6(s) - f_1(s)|$ and reaches absolute maximum $D = 0.0207995$ for $s = 0.684824$, meanwhile $D(0) = D'(0) = D''(0) = \lim_{s\to+\infty} D(s) = \lim_{s\to-\infty} D(s) = 0$. Therefore the difference between new and traditional perceptron characteristics is rather symbolic then dramatic and the way is open for the design of new artificial neural network.

4. Perceptron Radial Basis Function ANN

The previous two theorems motivate us to build up a new kind of artificial neural network with two types of processing neurons. The diversity of neuron types is not bad news for RBF network lovers.

Definition 1: Let $n \in \mathbf{N}$, $x_k, w_k \in \mathbf{R}$ for $k = 0, \ldots, n$, $x_0 = 1$. Then the function $y = \varphi(\mathbf{x}, \mathbf{w}) = \sum_{k=0}^{n} w_k x_k$ is called *linear neuron.*

Definition 2: Let $n \in \mathbf{N}$, $s_k \in \mathbf{R}$ for $k = 1, \ldots, n$. Then the function $G(\mathbf{s}) = \frac{1}{2^n}\prod_{k=1}^{n}\left(1 + \mathrm{sign}(s_k)\left(1 - \exp\left(-4s_k^2\right)\right)^{1/2}\right)$ is called *multiplicative perceptron.*

Now we can define the Perceptron Radial Basis Function ANN as a hierarchical artificial neural network with processing layers of two kinds.

Definition 3: Let $L \in \mathbf{N}$ be the number of layers. Let $N_k \in \mathbf{N}$ be the number of neurons in k^{th} layer of hierarchical ANN for $k = 1, \ldots, L$. Let the 1^{st} layer consist of input neurons. Let L^{th} layer be the output layer. Let $2j$ layer consists of linear neurons for $j = 1, \ldots, \lfloor L/2 \rfloor$. Let $2j+1$ layer consists of multiplicative perceptrons for $j = 1, \ldots, \lceil L/2 \rceil - 1$. Then the network is called a *Perceptron Radial Basis Function ANN* and denoted as $PRBF - N_1 - N_2 - \cdots - N_L$ or $PRBFL$.

As will be proven below, any linear ANN can be realized as $PRBF2$, any RBF network can be realized as $PRBF4$, any two-layer perceptron network can be

approximated as $PRBF3$, any three-layer perceptron network with linear output (MLL) can be approximated as $PRBF4$, and any three-layer perceptron network (MLP) can be approximated as $PRBF5$. The PRBF is a hierarchical ANN with incomplete connectivity of neighbor layers. Any multiplicative perceptron needs not operate on a complete set of neurons of the previous linear layer. The exact description of PRBF structure and parameters will be established via its matrix representation.

Definition 4: Let $L \in \mathbf{N}$. Let $N_1, \ldots, N_L$ be layer sizes of PRBF. Let $\mathbf{W}^{(2j-1)} \in \mathbf{R}^{N_{2j} \times (N_{2j-1}+1)}$ for $j = 1, \ldots, \lfloor L/2 \rfloor$. Let $\mathbf{W}^{(2j)} \in \{0;1\}^{N_{2j+1} \times N_{2j}}$ for $j = 1, \ldots, \lceil L/2 \rceil - 1$. Then $L-1$ tuple of matrices $\left(\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L-1)}\right)$ is called *matrix representation of PRBF.*

The interpretation of matrix rows is clear. The i^{th} row of matrix $\mathbf{W}^{(2j-1)}$ consists of weight vector for i^{th} linear neuron of $2j$ layer, which can be denoted as $\left(w_{i,0}^{(2j-1)}, \ldots, w_{i,N_{2j-1}}^{(2j-1)}\right)$. The i^{th} row of matrix $\mathbf{W}^{(2j)}$ consists of selecting vector for i^{th} multiplicative perceptron of $2j+1$ layer, which can be denoted as $\left(w_{i,1}^{(2j)}, \ldots, w_{i,N_{2j}}^{(2j)}\right)$. The matrix representation will help to describe PRBF structure, processing and learning.

Let $n, m \in N$ be the number of input and output neurons of linear layer. Let $\mathbf{x} = (x_0, \ldots, x_n)' \in \mathbf{R}^{n+1}$ be the input vector of linear layer with $x_0 = 1$, $\mathbf{y} = (y_1, \ldots, y_m)' \in \mathbf{R}^m$ be its output vector, and $\mathbf{W} \in \mathbf{R}^{m \times (n+1)}$ be its weight matrix. Then $\mathbf{y} = \mathbf{W}\mathbf{x}$ for every linear layer.

Let $n, m \in \mathbf{N}$ be the number of input and output neurons of multiplicative perceptron layer. Let $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{y} \in \mathbf{R}^m$, $\mathbf{W} \in \{0;1\}^{m \times n}$ be its input vector, output vector and connectivity matrix. Let $F : \{0;1\}^{m \times n} \times \mathbf{R}^n \to \mathbf{R}^m$ be a function, where $F_i(\mathbf{C}, \mathbf{s}) = \prod_{c_{i,k}=1} f_6(s_k)$ for $i = 1, \ldots, m$. Then $\mathbf{y} = F(\mathbf{W}, \mathbf{x})$ for every layer of multiplicative perceptrons.

The PRBF processing can be described due to its algorithm.

Algorithm 1:

Set input values into $\mathbf{x}$
Set *flag* $= 1$
For $j = 1, \ldots, L-1$ do:
    if *flag* $= 1$
    $\mathbf{x} = \mathbf{W}_j \mathbf{x}$
else
    $\mathbf{x} = F(\mathbf{W}_j, \mathbf{x})$
flag $= 1$ – *flag*
4. Produce vector of output values $\mathbf{y} = \mathbf{x}$

Formally, the PRBF network is function $\mathbf{y} = PRBF\left(\mathbf{x}, \mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L-1)}\right)$.

Supposing the training set of patterns $(\mathbf{x}_k, \mathbf{y}_k^*)$ for $k = 1, \ldots, M$, we can use least square method for learning the PRBF. Then the sum of squares

$$SSQ\left(\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L-1)}\right) = \sum_{k=1}^{M} \left\| \mathbf{y}_k^* - PRBF\left(\mathbf{x}_k, \mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L-1)}\right) \right\|^2$$

is subject of minimization, where $\|\ldots\|$ is Euclidean norm.

Definition 5: The minimization of $SSQ\left(\mathbf{W}^{(1)},\ldots,\mathbf{W}^{(L-1)}\right)$ for a given pattern set and topology of PRBF network is called *PRBF learning.*

The minimization of SSQ is a mixed integer non-linear programming task which is difficult to solve. In case of fixed interconnectivity matrices $\mathbf{W}^{(2)},\mathbf{W}^{(4)},\ldots$ the function $SSQ^*\left(\mathbf{W}^{(1)},\mathbf{W}^{(3)},\ldots\right)$ is continuous, smooth but multimodal function in general and the learning is also difficult. There are several possibilities how to earn PRBF:

- local optimization
- random shooting (Monte Carlo)
- stochastic gradient learning – backpropagation [3]
- differential evolution
- annealing
- combination of previous principles

The role of "good" initial structure and parameters of PRBF is cardinal. The relationships between PRBF, MLL, MLP, RBF and OLAM will help to find initial structure and weights of PRBF and then continue to the nearest local optimum of SSQ^*.

The PRBF network is able to realize any OLAM and RBF network and to approximate any MLL or MLP network. Useful properties of PRBF network are summarized in theorems 2–6.

Theorem 2: Any OLAM network can be realized as PRBF for $L = 2$.

Theorem 3: Let $n_I, n_H, n_0 \in \mathbf{N}$ be the number of input, hidden and output neurons of RBF network. Then the RBF network can be realized as PRBF for $L = 4$, $N_1 = n_I$, $N_2 = 2n_In_H$, $N_3 = n_H$, $N_4 = n_0$.

Theorem 4: Let $n_I, n_0 \in \mathbf{N}$ be the number of input neurons of two-layer perceptron network with logistic perceptrons (MLP 2). Then the network can be approximated as PRBF for $L = 3$, $N_1 = n_I$, $N_2 = N_3 = n_0$.

Theorem 5: Let $n_I, n_H, n_0 \in \mathbf{N}$ be the number of input, hidden and output neurons of three-layer perceptron (MLP 3). Then the network can be approximated as PRBF for $L = 5$, $N_1 = n_I$, $N_2 = N_3 = n_H$, $N_4 = N_5 = n_0$.

Theorem 6: Let $n_I, n_H, n_0 \in \mathbf{N}$ be the number of input, hidden and output neurons of threelayer perceptron with linear output neurons (MLL). Then the network can be approximated as PRBF for $L = 4$, $N_1 = n_I$, $N_2 = N_3 = n_H$, $N_4 = n_0$.

According to the previous theorems, PRBF network is able to realize any OLAM and RBF networks and approximate any MLP 2, MLP 3 and MLL networks with logistic perceptrons. Then PRBF network covers OLAM, RBF, MLP 2, MLP 3 and MLL networks and it is able to enrich their potential, as will be demonstrated in the experimental part.

5. Numerical Experiments

We have designed new testing environment in Matlab which allows testing of ANN learning. It is possible to add new ANN models and compare them with other ones. Both MLP and RBF networks can be learned via backpropagation algorithm [3]. PRBF network can be learned this way by using analytical gradient of PRBF. However, we prefer nongradient approach of local minimum searching. The weights of ANN are optimized by the application of fmincon function in Matlab, which employs the LSQ method in global minimum searching via repeated local optimization from random initial points. Following values are recorded:

- ni – nuber of input neurons
- nh – number of neurons in hidden layer
- no – number of output neurons
- nw – number of weights
- df – degrees of freedom
- ssq – sum of squares of ANN variances
- sy – model error as $sy = \sqrt{\frac{ssq}{df}}$
- ne – average number of SSQ evaluation from 100 runs

The improvement of model error was tested in the case of PRBF network related to the RBF one. The PRBF learning was based on optimum RBF weights. Based on Theorem 3, the weights of RBF were converted to equivalent weights of PRBF. Every input x_k of original RBF neuron is then split into two inputs $y_k = \frac{x_k - w_k}{2\sqrt{2}\sigma}$, $y_{k+n_I-1} = \frac{w_k - x_k}{2\sqrt{2}\sigma}$ of PRBF neuron. The optimum parameters of PRBF were searched in the neighborhood of this initial estimate in constrained space with 10% tolerance as local LSQ optimum, of course.

The tests were performed on ANN time series prediction task. Data set of annual number of sunspots (sunspots.dat) is freely available in the Matlab environment. Only the MLP with characteristics $f_1(s) = \frac{1+\tanh 2s}{2}$, RBF with characteristics $g_1(s) = \frac{1}{4}\exp\left(-4s^2\right)$ and PRBF with three input neurons and single output neuron were tested and compared. Various numbers of hidden neurons up to 5 were used for the study of MLP, RBF and PRBF properties.

6. Results

Results of MLP, RBF and PRBF learning were collected in the Tab. I. The optimum structures with the smallest model error (within the type of ANN) were MLP 3-4-1, RBF 3-3-1 and PRBF 3-4-1. Nevertheless, the PRBF network had the better performance than the MLP or RBF one. The learning rate was measured as a mean number of SSQ evaluations during repeated local optimization. As seen in Tab. I, the time complexity (ne) increases with the number of MLP, RBF and PRBF. The time complexity of PRBF learning is higher than the time complexity of MLP and RBF learning.

ANN model	*ni*	*nh*	*no*	*nw*	*df*	*sy*	*ssq*	*ne*
MLP	3	1	1	6	278	0.144562	5.809707	75
MLP	3	2	1	11	273	0.128509	4.508461	621
MLP	3	3	1	16	268	0.126624	4.297032	874
MLP	3	4	1	21	263	**0.125553**	4.145817	990
MLP	3	5	1	26	258	0.126550	4.131851	1021
RBF	3	1	1	6	278	0.132752	4.899236	36
RBF	3	2	1	11	273	0.124625	4.240088	571
RBF	3	3	1	16	268	**0.122964**	4.051967	809
RBF	3	4	1	21	263	0.123880	4.036092	924
RBF	3	5	1	26	258	0.124729	4.011853	986
PRBF	3	1	1	8	276	0.117532	3.812577	754
PRBF	3	2	1	15	269	0.117688	3.725774	834
PRBF	3	3	1	22	262	0.116791	3.573745	896
PRBF	3	4	1	29	255	**0.115388**	3.395183	970
PRBF	3	5	1	36	248	0.115343	3.299403	1089

Tab. I *Results of MLP, RBF and PRBF learning.*

7. Conclusions

Three types of ANN (MLP, RBF, PRBF) were learned to be the best predictors of sunspot number from last three-year history. The results show that PRBF was the best model in the case of model error minimization. The PRBF network has more weights then MLP or RBF with the same number of nonlinear neurons, which reduces the degrees of freedom. However, this effect is included in the model error calculations, and thus we recommend the PRBF network as a very efficient tool for data modeling. The learning of PRBF network has to be preceded with RBF learning, which is a good generator of initial PRBF weight estimate.

Acknowledgements

The paper was created under the support of grant No. SGS11/165/OHK4/3T/14 CTU in Prague.

References

[1] Haykin S.: Neural Networks, Macmillan, New York, 1994.

[2] Poggio T.: Networks for Approximation and Learning, Proceedings of the IEEE, 78, 1990, pp. 1481-1497.

[3] Rumelhart D. E., Hinton G. E., Williams R. J.: Learning Representations by Back-Propagating Error. Nature, 323, 1986, pp. 533-536.