

Weighted Radial Basis Functions for Improved Pattern Recognition and Signal Processing

Leonardo M. Reyneri

Dipartimento di Ingegneria dell'Informazione, Univerità di Pisa
Via Diotisalvi 2, 56126 Pisa, ITALY
e.mail lmr@iet.unipi.it

Abstract

The paper describes an improved version of the Radial Basis Function algorithm, which integrates the advantages of Multi-Layer Perceptrons and Radial Basis Functions alone. The proposed paradigm is more general in nature, since it has the other two as particular subcases. It finds applications in several pattern recognition and classification tasks. Furthermore it can also be used as a method to map Fuzzy Inference Systems on Artificial Neural Networks.

1 Introduction

Multi-Layer Perceptrons (**MLPs**) [1] and Radial Basis Functions (**RBFs**) [2] are two Neural computing paradigms widely used in several applications, ranging from pattern recognition, to industrial control, Neuro-Fuzzy emulation, function approximation, etc. Each of the two paradigms has its own advantages and drawbacks, which are not discussed here. Fuzzy Systems (**FSs**) [2] represent another computational paradigm widely used in several intelligent controller applications [3].

Although apparently the three paradigms mentioned above are very different from each other, they can be unified into what has been called *Weighted Radial Basis Functions (WRBFs)*, as described in this letter.

The proposed algorithm is associated to a learning rule which is also a combination of MLP's gradient descent and RBF's learning methods. Sections 2 and 3 describe the WRBF algorithm and its learning rule, respectively, while Section 4 describes a method for the initialization of WRBF parameters. Finally, Section 5 briefly discusses some application.

2 Weighted Radial Basis Functions

For the proposed algorithm, each neuron $j \in [1 \dots M]$ is associated to a pair of vectors, namely a *location center* $\vec{C}^j = \{c_1^j, c_2^j, \dots, c_N^j\}$ (as for RBFs) and a *weight vector* $\vec{W}^j = \{\Theta^j, w_1^j, w_2^j, \dots, w_N^j\}$ (as for MLPs). The output y^j of the neuron is a function of the input vector $\vec{X} = \{x_1, x_2, \dots, x_N\}$:

$$y^j = \mathcal{H}_n^{F(z)}(\vec{X}; \vec{C}^j, \vec{W}^j) \quad (1)$$

where $\mathcal{H}_n^{F(z)}(\cdot)$ is the characteristic of a *Weighted Radial Basis Function of order n*, which is defined as:

$$\mathcal{H}_n^{F(z)}(\vec{X}; \vec{C}^j, \vec{W}^j) \triangleq F\left(\Theta^j + \frac{1}{N} \sum_{i=1}^N \mathcal{D}_n(x_i - c_i^j) \cdot w_i^j\right), \quad (2)$$

where the factor $\mathcal{D}_n(x_i - c_i^j)$ is a function of the distance between the i -th component of the input vector \vec{X} and of the location center \vec{C}^j , respectively:

$$\mathcal{D}_n(x_i - c_i^j) \triangleq \begin{cases} (x_i - c_i^j) & \text{for } n = 0 \\ |x_i - c_i^j|^n & \text{for } n > 0 \end{cases} \quad (3)$$

$F(\cdot)$ is one of several possible monotonic *activation functions* such as, for instance:

$$F(z) \triangleq \begin{cases} \frac{1 - e^{-z}}{1 + e^{-z}} & (\text{sigmoidal}) \\ e^{-z} & (\text{exponential}) \\ z & (\text{linear}) \end{cases} \quad (4)$$

One or more WRBF layers can be cascaded to build a Multi-Layer WRBF, by connecting all the outputs of a layer to the inputs of the next one, as for MLPs [1]. For instance:

$$\vec{Y} = \mathcal{H}_{n_2}^{F_2} \left(\mathcal{H}_{n_1}^{F_1} \left(\vec{X}; \vec{C}_1, \vec{W}_1 \right); \vec{C}_2, \vec{W}_2 \right) \quad (5)$$

where $\vec{Y} = \{y_1^j, y_2^j, \dots, y_N^j\}$ is the vector of outputs, while \vec{C}_k , \vec{W}_k , n_k and F_k are the location centers matrices, the weight matrices, the order and the activation function of the k -th network layer.

Note that the standard RBF and MLP paradigms can be seen as two cases of WRBFs, while FSs can be approximated by an appropriate WRBF:

- **MLPs** [1] are equivalent to $\mathcal{H}_0^{\text{sigm}} \left(\vec{X}; \vec{0}, \vec{W}^j \right)$ with sigmoidal activation function, where $\vec{0}$ is a vector of 0s.
- **RBFs** [2] are equivalent to $\mathcal{H}_2^{\text{exp}} \left(\vec{X}; \vec{C}^j, \vec{1} \right)$ with exponential activation function, where $\vec{1}$ is a vector of 1s. With weight vector components equal to one, the input/output characteristic of the neuron is equivalent to that of a RBF [2], since the exponential function (4) behaves as a gaussian, due to the exponent $n = 2$ in (2), (3).
- **FSs** [2] can be approximated by a two-layer RBF. This statement can be explained by giving an example of a set of Fuzzy inference rules:

IF (x1 IN R1A) AND (x2 IN R2A) AND ... (xN IN RNA) THEN SA
IF (x1 IN R1B) AND (x2 IN R2B) AND ... (xN IN RNB) THEN SB

where R1A, R2A, ... are appropriate *membership functions* [2]. In case of *bell-shaped* (i.e. gaussian-like) functions, the above rules can be computed analytically as:

$$y_{RBF} \approx T_A \cdot \min \left\{ \mathcal{R}_1^A(x_1), \mathcal{R}_2^A(x_2), \dots, \mathcal{R}_N^A(x_N) \right\} + T_B \cdot \min \left\{ \mathcal{R}_1^B(x_1), \mathcal{R}_2^B(x_2), \dots, \mathcal{R}_N^B(x_N) \right\} + \dots \quad (6)$$

where T_A and T_B are the *centers of gravity* of membership functions SA and SB, respectively, while $\mathcal{R}_i^A(x_i)$ and $\mathcal{R}_i^B(x_i)$ are the membership value of input x_i to the rules RiA and RiB, respectively. Under the assumption that (valid for several bell-shaped functions):

$$\mathcal{R}_i^j(x_i) \approx e^{(x_i - c_i^j)^2 \cdot w_i^j}, \quad (7)$$

equation (6) can be approximated by:

$$y_{RBF} \approx T_A \left(e^{(x_1 - c_1^1)^2 \cdot w_1^1} \cdot e^{(x_2 - c_2^1)^2 \cdot w_2^1} \cdot \dots \right) + T_B \left(e^{(x_1 - c_1^2)^2 \cdot w_1^2} \cdot e^{(x_2 - c_2^2)^2 \cdot w_2^2} \cdot \dots \right) + \dots \quad (8)$$

N	Membership function normalized width	RMS error (%)
2	0.5	14.9
	0.2	6.0
	0.1	3.0
3	0.5	18.0
	0.2	4.6
	0.1	1.6

Table 1: RMS errors approximating a Fuzzy System with a WRBF. The width of the bell-shaped membership function is expressed as a fraction of the size of the input universe.

where the operator “min” in formula (6) has been approximated by a multiplication. Such an approximation introduces a small error, as indicated in Tab. 1, for $N = 2$ and $N = 3$ inputs. This error is also function of the *normalized width* of the membership function. Finally, formula (8) can be modified as:

$$y_{RBF} \approx \sum_{k=1}^M \left(T_k \cdot e^{\sum_{i=1}^N (x_i - c_i^k)^2 \cdot w_i^k} \right), \quad (9)$$

which is the characteristic of a two-layer WRBF (from (2), (3), and (4)):

$$\mathcal{H}_0^{\text{lin}} \left(\mathcal{H}_2^{\text{exp}} \left(\vec{X}; \vec{C}, \vec{W} \right); \vec{0}, \vec{T}^1 \right), \quad (10)$$

where \vec{C} and \vec{W} are the *center* and *weight matrices*, respectively. In other words, this is a WRBF on the first layer plus a MLP with linear activation function on the second layer.

3 Learning Rule

An ad-hoc learning rule has been developed for WRBFs, as the “combination” of learning rules of MLPs and RBFs. The proposed rule is of the supervised type, therefore a *target pattern* \vec{T}_p is associated to each *input pattern* \vec{X}_p in the *training set*. Furthermore, an *error function* is defined as:

$$E(p) = \sum_{j=1}^N \left(t_p^j - y_p^j \right)^2, \quad (11)$$

where y_p^j and t_p^j are the j -th network output and the corresponding target, respectively, when the p -th pattern is presented to the input.

3.1 Weight Vectors

For the weight vectors, a *gradient descent method* is used, as for MLPs [1], where the corrections to the weight vectors are made proportional to the gradient of the error function ∇E , by means of a variable *learning rate* η . Therefore, from (2) and (4):

$$\Delta w_i^j = \eta \delta^j F'(z) D_n \left(x_i - c_i^j \right), \quad (12)$$

where $\delta^j = (t^j - y^j)$ and the factor $D_n(x_i - c_i^j)$ substitutes the x_i in the MLP’s learning rule [1]. In the case of a Multi-Layer WRBF, on the other layers except for the last one:

$$\delta_{(k-1)}^i = \sum_{j=1}^M \left(\delta_{(k)}^j F'_{(k)}(z^j) w_{(k)i}^j \beta_{(k)} \right) \quad (13)$$

where $\delta_{(k-1)}^i$ is the error term for the previous layer, while:

$$\beta = \begin{cases} 1 & \text{for } n = 0 \\ n \cdot (x_i - c_i^j)^{(n-1)} & \text{for } n > 0, \text{ even} \\ n \cdot (x_i - c_i^j)^{(n-1)} \cdot \text{sgn}(x_i - c_i^j) & \text{for } n > 0, \text{ odd} \end{cases} \quad (14)$$

3.2 Location Centers

As for standard RBFs, learning of location centers takes place according to an auto-organizing paradigm [1], which can be described as follows:

1. a pattern \vec{X}_p from the training set is applied to the WRBF inputs;
2. the neuron with the highest output activation is declared the *winner*. For classification purposes, neurons can be grouped into as many aggregates as there are classes. In this case, the search for the winner is limited to the aggregate logically associated to the input pattern;
3. the location center of the winner \vec{C}^j (and optionally of a limited neighborhood) is trained according to:

$$\Delta c_i^j = \gamma(x_i - c_i^j), \quad (15)$$

where γ is an appropriate learning coefficient. The value of γ can vary with the epochs of learning and with the distance from the winner (if a neighborhood exists);

4. the algorithm is repeated for each training pattern.

After a sufficient number of iterations, the location centers and weight vectors tend to converge to the good solution. Obviously, learning of weights and location centers affect each other, therefore the order in which they are performed can influence on the performance of the network.

4 Weight and Center Initialization

This section discusses on a pre-training initialization of weights and location centers, which has proven useful in several pattern classification applications. Note that the initialization proposed is not applicable to all cases, and is not a fundamental step of the WRBF algorithm. In fact, in several applications a random weight initialization (or other types of initialization) may perform better. The major advantage of the proposed method is its simplicity and ease of implementation.

A pattern classifier implemented as a two-layer WRBF is considered here as an example:

$$\mathcal{H}_0^{\text{lin}} \left(\mathcal{H}_2^{\text{exp}} \left(\vec{X}; \vec{C}, \vec{W} \right); \vec{0}, \vec{T} \right), \quad (16)$$

In the first layer, a group of M neurons is associated to each class, while in the second layer there is just one neuron per class, which is connected only to the outputs of the corresponding group of neurons in the first layer, as shown in Fig. 1.

The proposed initialization applies only to the first layer, as in the second layer all the weights are equal either to 0 (unconnected inputs) or to 1 (connected inputs).

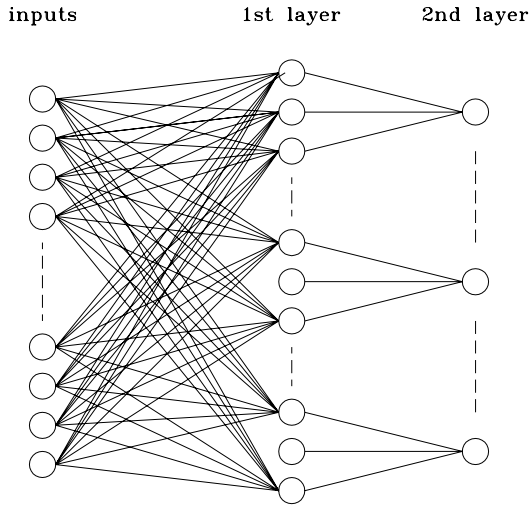


Figure 1: Simplified diagram of the WRBF pattern classifier

4.1 Initialization of Location Centers

All the patterns of a class are subdivided into a given number of subsets and the following value is assigned to the j -th location center:

$$\vec{C}^j = \frac{1}{R(P^j)} \sum_{p \in P^j} \vec{X}_p, \quad (17)$$

where P^j and $R(P^j)$ are the j -th subset and the corresponding number of components, respectively. This means that, for each subset, its associated location center will contain the *average* of the patterns of the subset, which is often a vector roughly representative of the subset elements.

Two methods have been considered to subdivide the reference patterns into the subsets P^j . In a first method (A), the patterns are spread randomly into the subsets, while the second method (B) is composed of two phases: in the first phase, the first pattern is assigned to the first subset, then the pattern which is farthest from the first one (under the metric (3)) is assigned to the second subset, then the pattern which is farthest away from the first two is assigned to the third, and so on, until all the subsets contain one pattern. Then, in the second phase, each one of the remaining patterns is assigned to the subset which contains the patterns closest to it.

The proposed method has been tested on a traditional RBF for the recognition of handwriting [4], where it has provided good results.

4.2 Initialization of Weight Vectors

The following method for the initialization of WRBF weights is tightly related to the initialization of the location centers proposed in Section 4.1. Once the reference patterns have been subdivided into the subsets, and the location centers computed (from (17)), the corresponding weight vectors can be initialized according to the following formula:

$$w_i^j = e^{\left(-\frac{\sigma_i(P^j)}{\rho}\right)} \quad (18)$$

where $\sigma_i(P^j)$ is the standard deviation (or alternatively the *entropy* [5]) of the i -th input in the subset P^j , while ρ is an appropriate parameter. This method assigns a larger weight to those pixels which have a lower standard deviation (or entropy), and therefore are more significant for the classification task, and less to the others, as expected.

5 Applications and Advantages

The proposed algorithm finds several applications, some of which have already been mentioned previously. It has been deeply tested in several practical cases [4, 6]. Due to space limitations, this section only discusses the relevant advantages of the proposed paradigm to two relevant application areas.

Pattern Classification: RBFs are known to provide interesting advantages with respect to MLPs, especially for what training concerns. Adding a different weight to each input improves learning performance, especially when dealing with noisy, blurry and not-well defined patterns, as can be found in a lot of practical problems (e.g. handwriting [4] and speech recognition).

Intelligent Control: Fuzzy and Neural controllers [3, 6] have different advantages and drawbacks. WRBF can take the advantages of both of them, in the sense that it allows the design of *learning controllers* by either human-friendly Fuzzy Inference methods (therefore reusing the know-how of human experts) or more computer-friendly Neural Learning methods (therefore learning from samples measured directly from the controlled plants). The similarity between WRBFs and FSs allows a straightforward learning of Neural Networks from Fuzzy Controllers and, viceversa, the easy translation into human-readable Fuzzy rules of trained WRBF Neural Networks.

6 Conclusion

This paper has proposed a new Neuro-Fuzzy paradigm, which derives from the combination of Multi-Layer Perceptrons and Radial Basis Functions, and which approximates Fuzzy Inference Systems. A supervised learning rule and a weight initialization procedure have also been developed. The proposed method has proven to be useful in several pattern classification tasks and for the efficient implementation of Neuro-Fuzzy controllers. An important advantage of WRBF is that it represents a unification algorithm between MLPs, RBFs, and FSs, therefore it is a link between two large categories of applications, namely those dominated by Fuzzy and Neural approaches, respectively.

Acknowledgments

The author wishes to thank Mr. Enrico Rossi for the useful discussions which led to the method described here.

References

- [1] P.D. Wasserman, "Neural Computing: Theory and Practice", *Van Nostrand Reinhold*, New York, 1989.
- [2] P.D. Wasserman, "Advanced Methods in Neural Computing", *Van Nostrand Reinhold*, New York, 1993.
- [3] D.A. White and D.A. Sofge, "Handbook of Intelligent Control", Van Nostrand Reinhold, New York, 1992.
- [4] B. Lazzarini, F. Marcelloni, L.M. Reyneri, E. Rossi, L. Schiuma. "Il Sistema BEATRIX per il Riconoscimento Automatico di Testi Manoscritti", *Proc. of AICA 94*, Palermo, September 1994, pp. 1303-1318.
- [5] L.W. Cough II, "Digital and Analog Communication Systems", Maxwell MaxMillan International Editions, New York, 1989.
- [6] L.M. Reyneri, M. Chiaberge, L. Zocca, "CINTIA: A Neuro-Fuzzy Real Time Controller for Low Power Embedded Systems", in *Proc. of MICRONEURO 94*, Torino (I), IEEE Computer Society Press, September 1994, pp. 392-404.