

Performance of Weighted Radial Basis Function Classifiers

L.M. Reyneri[†], M. Sgarbi^{††}

[†] Dipartimento di Elettronica - Politecnico di Torino - ITALY

^{††} Via Aquilaia, 302 - 58054 Scansano (GR) - ITALY

Abstract

This paper describes Weighted Radial Basis Functions, a neuro-fuzzy unification algorithm which mixes Perceptrons and Radial Basis Functions. The algorithm has been tested as a pattern classifier in practical applications. Its performance are compared against those of other neural classifiers. The proposed algorithm has performance comparable or better than other neural algorithms, although it can be trained much faster. It can also act as a neuro-fuzzy unification algorithm.

1 Introduction

Most traditional neural pattern classifiers [1, 2, 3, 4] are based on either Multi Layer Perceptrons (MLPs) or Radial Basis Functions (RBFs) [1]. Both algorithms have several advantages and drawbacks [5], some of which are shortly described below.

MLPs are *correlation-based* algorithms: each neuron output is based on a non-linear correlation between the input vector \vec{X} and a *weight vector* \vec{W}^j associated with neuron j . Correlation is used since it is an operator which maximizes the output when \vec{X} and \vec{W}^j are most similar, *provided that input magnitude $\|\vec{X}\|$ is constant* [5]. Unfortunately, when that is not constant (as it often happens in classification tasks), the higher is $\|\vec{X}\|$ the higher is the neuron output, even if \vec{X} and \vec{W} are not similar. This requires either to normalize the input vector (when input magnitude is immaterial for the classification task), or to use a more complex network (namely, two layers instead of just one).

Another problem typical of MLPs trained using either Back-propagation or Hebb rules is that weight corrections are proportional to the input value, therefore weights associated with low input values cannot easily be trained.

RBFs are *distance-based* algorithms: each neuron output is a non-linear function of the distance between the input vector \vec{X} and a *center vector* \vec{C}^j associated with neuron j . The distance is a metric which is used as it maximizes the output when \vec{X} and \vec{C}^j are most similar, *independent of input magnitude $\|\vec{X}\|$* (this is the major difference with respect to MLPs).

Yet, RBF networks may suffer from the *dimensional problem*, which arises when trying to sum up different inputs which do not have the same physical dimension (namely, an heterogeneous input space). Another problem is that decision boundaries, which are always spherical, may not be optimal in several classification and function approximation applications.

Note that, when the input magnitude is constant, MLP and RBF neurons can be made almost identical to each other [5]. That happens, for instance, either when input vectors are normalized, or with binary inputs, or with images taken from a camera with automatic brightness compensation.

One major problem which arises in both MLPs and RBFs can be explained through an example: in MLP classifiers trained using supervised learning rules [1], it often happens that there are several input elements which are associated with large weight values, although they are immaterial for the required classification (for instance, all the pixels in the white or black

background of an image). Such weights cause a higher sensitivity to noise for a given network accuracy. This problem also exists in RBFs, which give the same (unit) weight to all inputs, therefore it is often less critical than in MLPs, yet still present.

In practice it would be desirable to have lower weight values associated with all those inputs which are less significant for the classification purpose. This is perhaps the major reason for the development of the Weighted Radial Basis Functions algorithm (**WRBFs**) [6] described below. WRBFs also solve other drawbacks of traditional neural algorithms, therefore provide improved classification performance, as described in section 3

2 Weighted Radial Basis Functions

This section briefly describes the WRBF algorithm and shows that other neural and fuzzy algorithms are just sub-cases. It can also be used as a neuro-fuzzy unification algorithm [5, 6].

A WRBF neuron is associated with a set of parameters: an *order* $n \in \mathcal{R}$, defining the neuron's *metric*; a *weight vector* \vec{W}^j , deriving from the weight vector of MLPs; a *center vector* \vec{C}^j , deriving from the center vector of RBFs; a *bias* Θ^j , deriving from MLPs; an *activation function* $F(z)$, common to both types of neurons. The mathematical model of a WRBF neuron of order n is:

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

where the *distance function* $\mathcal{D}_n(\cdot)$ is given by:

$$\mathcal{D}_n(x_i - c_i) \triangleq \begin{cases} (x_i - c_i^j) & \text{for } n = 0 \\ |x_i - c_i^j|^n & \text{for } n \neq 0 \end{cases} \quad (2)$$

and $F(z)$ is a generic *activation function*, although in most cases monotonic functions are used, such as [5]:

$$F(z) = \begin{cases} \frac{F_{\min}e^{-z/\gamma} + F_{\max}e^{+z/\gamma}}{e^{-z/\gamma} + e^{+z/\gamma}} & (\text{generalized sigmoid}) \\ F_{\min} + (F_{\max} - F_{\min})e^{-|z/\gamma|^m} & (\text{generalized exponential}) \end{cases} \quad (3)$$

where γ is an appropriate *width (normalization) factor*. Many neural and fuzzy algorithms can be re-conducted to WRBF:

- an *MLP* neuron is identical to a WRBF neuron of order 0, provided that $\vec{C}^j = 0$, and $F(z)$ is a generalized sigmoid. It is worth noting that introducing an additional vector \vec{C}^j into MLPs is equivalent to adding a *bias* $-c_i^j$ to each input x_i . This reduces a drawback of most back-propagation algorithms, which cannot correct weights connected to inputs which are zero (or very small).
- an *RBF* neuron is identical to a WRBF neuron of order n (typically, $n = 2$), provided that $\vec{W}^j = 1$, and $F(z)$ is a generalized exponential with $m = 1$. Note that multiplying each distance component $|x_i - c_i^j|^n$ by a factor w_i both solves the dimensional problem and produces elliptical decision boundaries, which may give better performance than spherical ones.
- WRBF neurons can also be made identical to several types of *fuzzy rules* by properly choosing their parameters [5].

2.1 Generalized learning rule

A generalized learning rule has also been developed for the WRBF algorithm. It is based on the gradient descent method, which delta rules [1] are based on. It can be shown [5] that, applying gradient descent methods to WRBF neurons includes the delta, back-propagation, and Kohonen's self-organizing, as sub-cases.

By applying the concept of *gradient descent* of a *quadratic error function* to the weight vector \vec{W}^j , the center vector \vec{C}^j , and the bias Θ^j , the generalized learning rule for WRBF neurons becomes (for $n \neq 0$):

$$\Delta w_i^j = -\eta_W \cdot (y^j - t^j) \cdot \frac{dF^j(z)}{dz} \cdot \mathcal{D}_n(x_i - c_i^j) \quad (4)$$

$$\Delta \Theta^j = -\eta_\Theta \cdot (y^j - t^j) \cdot \frac{dF^j(z)}{dz} \quad (5)$$

$$\Delta c_i^j = \eta_C \cdot (y^j - t^j) \cdot \frac{dF^j(z)}{dz} \cdot w_i^j \cdot \left[\mathcal{D}_{(n-1)}(x_i - c_i^j) \cdot \text{sgn}(x_i - c_i^j) \right] \quad (6)$$

$$\delta x_i = - \sum_j (y^j - t^j) \cdot \frac{dF^j(z)}{dz} \cdot w_i^j \cdot \left[\mathcal{D}_{(n-1)} \cdot (x_i - c_i^j) \cdot \text{sgn}(x_i - c_i^j) \right] \quad (7)$$

where η_W , η_C , and η_Θ are appropriate *learning coefficients*, while $F(z)$ is the activation function (see (3)). The last formula applies only to multi-layer networks and provides the correction vector to be back-propagated [1] (*generalized back-propagation algorithm*).

A particular case holds for $n = 0$: formulae (4) and (5) remain the same while, in (6) and (7), the terms between square brackets are removed. See [5] for further details on the unsupervised generalized learning rule.

2.2 Weight and Center Initialization

This section discusses on a pre-training initialization of weights \vec{W}^j and centers \vec{C}^j , which has proven useful in several pattern classification applications (see, for instance section 3). Initialization provides weight and center values which are often close enough to the optimal value, therefore the following learning phase is much shorter. Initialization alone is often sufficient to get a working (although non-optimal) classifier, as shown in plots 1 and 2. Note that the initialization proposed here is not applicable to all cases, and is not a fundamental step of the WRBF algorithm. A random weight initialization (or other types of initialization) may perform better in several other applications. The major advantage of the proposed method is its simplicity and ease of implementation.

The initialization algorithm applies primarily to pattern classifiers implemented as two-layer WRBFs. In the first layer, a group of M neurons is associated with each class, while the second layer has just one neuron per class, connected only to the outputs of neurons of the same class. Initialization applies to the first layer, as in the second layer weights are initially set to 1.

The training set \mathcal{T} is first subdivided into M subsets \mathcal{P}^j per each class. A simple though effective method to subdivide patterns 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 (2)) 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.

After all patterns have been subdivided, WRBF centers and weights are initialized with the

following values:

$$\vec{C}^j = \frac{1}{N(\mathcal{P}^j)} \sum_{p \in \mathcal{P}^j} \vec{X}_p, \quad (8)$$

$$w_i^j = \tau \cdot \frac{\sigma_i(\mathcal{T})}{\sigma(\mathcal{T})^{(n+1)}} \cdot e^{\left(-\rho \frac{\sigma_i(\mathcal{P}^j)}{\sigma_i(\mathcal{T})}\right)} \quad (9)$$

where \mathcal{P}^j and $N(\mathcal{P}^j)$ are the j -th subset and the corresponding number of elements, respectively, while $\sigma_i(\mathcal{P})$ (respectively, $\sigma(\mathcal{P})$) is the standard deviation of the i -th input (respectively, of all the inputs) in the set \mathcal{P} , and ρ and τ are appropriate parameters. When $\rho = 0$, all weight matrices \vec{W}^j become identical to each other, therefore reducing classification time (by substituting $x'_i = \sqrt{w_i}x_i$, $c_i^j = \sqrt{w_i}c_i^j$ and $w_i^j = 1$ into (1)).

For each subset, the associated location center \vec{C}^j will contain the *average* of the patterns of the subset, which is often a vector roughly representative of the subset elements. Initialization assigns a larger weight to those pixels which have a lower intra-class standard deviation $\sigma_i(\mathcal{P}^j)$, and therefore are more significant for the classification task, and less to the others, as expected.

3 Performance Comparison

This section compares the performance of two cases of the WRBF algorithms with more traditional MLPs and RBFs. Performance are evaluated on a non-trivial practical example, which has been tested on a commercially available system. The system has to classify paper documents according to a colored logo printed on a well known area of the document. The logo is acquired through a color camera, sampled in four different areas of the image (windows), 32 samples per each window, three colors per sample, for a total of 384 input elements, 256 gray levels per each input. Documents must be classified into 20 different classes. Training and test sets are composed of 364 and 727 patterns, respectively (namely about 18 and 36 pattern per class, respectively). Initializations, training and simulations have all been executed on the **FANNLAB** neuro-fuzzy simulator.

Plots 1 to 5 compare the performance of four networks: an MLP, an RBF, and two WRBFs, of order 1 and 2, respectively. All the networks use only 96 inputs (only one of the four windows is used) and have exactly the same size, namely one layer, 96 inputs, 20 neurons. Note that, as all input patterns have approximately the same amplitude ($\vec{X} \approx \text{const}$), the linear separation problem typical of MLPs does not cause problems [5], therefore a one-layer MLP is sufficient (although it may have poorer performance). The MLP uses a sigmoidal activation function, while RBF and the two WRBFs use an exponential activation function. In both cases the outputs range between 0 and 1.

During recognition, the networks accepts a pattern *iff* one and only one output is higher than a given *upper decision threshold* ϕ_H , while all the other outputs are below a *lower decision threshold* ϕ_L . The network rejects any other pattern. The thresholds have been chosen to have no errors, as the cost of errors in the proposed application is very high.

Some performance comparisons are based on a *noise sensitivity plot*, which expresses in graphical form the rejection rate versus the standard deviation σ_n of the white Gaussian noise added to each input.

Plot 1 compares the performance of three networks (MLP is excluded), which have been initialized as described in section 2.2, but not trained. The decision thresholds are $\phi_L = 0.2$, $\phi_H = 0.5$, while $\rho = 1$ and $\gamma = 1$. Initialization performs well for both WRBF2 and RBF, while it shows poor performance with WRBF1.

Plot 2 is similar to plot 1 but applies to the same networks after training (MLP is trained starting from random initial values). Performance of the three networks improve significantly

within less than 100 training epochs (sometimes 10-20 epochs are enough), while MLP requires a much longer and more critical training phase (often more than 1000 epochs). WRBF2 and MLP show comparable performance, while RBF show the worst performance, and MLP suffers from a much slower training phase.

Plot 3 shows rejection rate versus decision thresholds σ_L and σ_H . When σ_L and σ_H approach to each other the error rate (not shown on the plot) increases.

Plot 4 shows rejection rate versus normalized image brightness. All networks except MLP are rather sensitive to brightness variations of more than 2-5%. Sensitivity is reduced significantly by removing from the input patterns the average image brightness. This gives the performance shown in plot 5.

Plot 6 is similar to plot 2 but applies to the four networks with all 384 inputs (namely, all the four windows are used). In this case network architecture is more complex: three layers, 384 inputs, 240, 120 hidden units and 20 outputs.

It can be seen that WRBF2 and MLP show comparable performance, while RBF show the worst performance. In many cases RBF reject about five times as many pattern as a WRBF2 and an MLP, while WRBF1 rejects about twice as many patterns as a WRBF2. Error rate is zero in all cases.

WRBFs have the advantage with respect to MLPs of requiring a much faster training phase, especially when updating the training set by modifying the patterns of only one or few classes. This advantage becomes even more significant when network size increases, as training time of MLPs increase significantly, while training time of WRBFs may remains constant. In many applications, training can be avoided.

On the other hand, WRBFs have the drawback of requiring more memory for weight/center storage, and one more operation per each synopsis. The effects of this drawback can be reduced significantly by properly storing weight matrices and implementing the algorithm (see an example in section 2.2).

References

- [1] S. Haykin, "Neural Networks: A Comprehensive Foundation", *Mc Millan College Publishing Company*, New York, 1994.
- [2] R.M. Bozinovic and S.N. Srihari, "Off-line cursive script word recognition", *IEEE Trans. on PAMI*, vol. 11, pp. 68-83, Jan. 1989.
- [3] M. Costa, E. Filippi and E. Pasero, "Multi-Layer Perceptron Ensembles for Pattern Recognition: Some Experiments", *Proc. of European Symposium on Artificial Neural Networks*, ESANN 94, Bruxelles, April 1994.
- [4] D. Wettschereck and T. Dietterich, "Improving the Performance of Radial Basis Function Networks by Learning Center Locations", in *Advances in Neural Information Processing System 3*, Morgan Kaufmann, pp.1133-1140, 1992.
- [5] L.M. Reyneri, "Unification of Neural and Fuzzy Computing Paradigms", in *Proc. of AT-96, 1-st Int'l Symposium on Neuro-Fuzzy Systems*, Lausanne, August 1996.
- [6] L.M. Reyneri, "Weighted Radial Basis Functions for Improved Pattern Recognition and Signal Processing", *Neural Processing Letters*, Vol. 2, No. 3, May 1995, pp. 2-6.

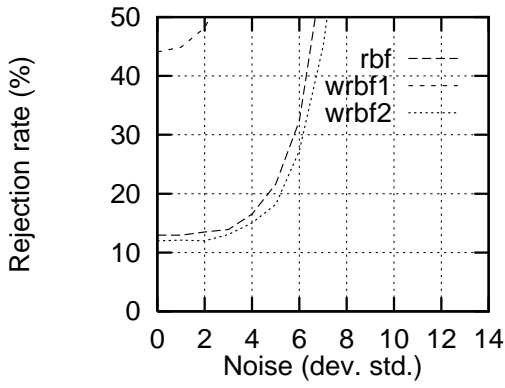


Figure 1: Noise sensitivity of an RBF and two WRBFs, for initialized but untrained networks, $\sigma_L = 0.2$, $\sigma_H = 0.5$.

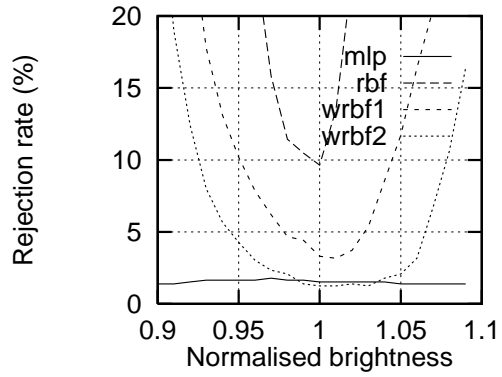


Figure 4: Rejection rate of an MLP, an RBF and two WRBFs versus normalised input brightness, without subtracting the average brightness.

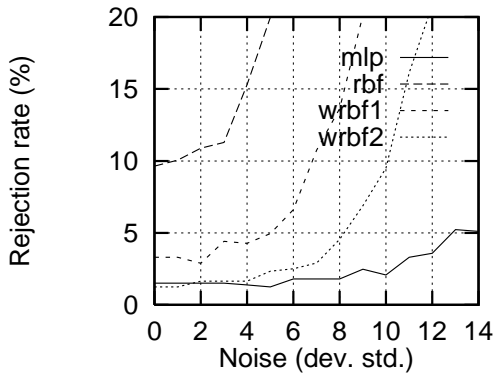


Figure 2: Noise sensitivity of an MLP, an RBF, and two WRBFs, after training.

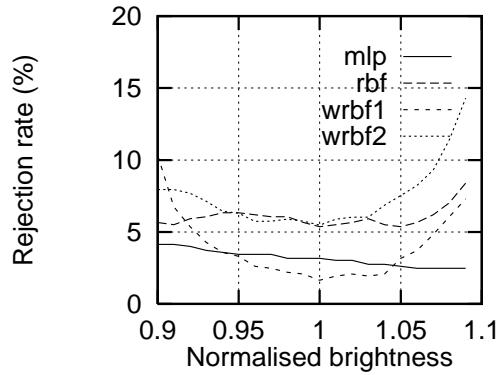


Figure 5: Rejection rate of an MLP, an RBF, and two WRBFs versus normalised input brightness, after subtracting the average brightness.

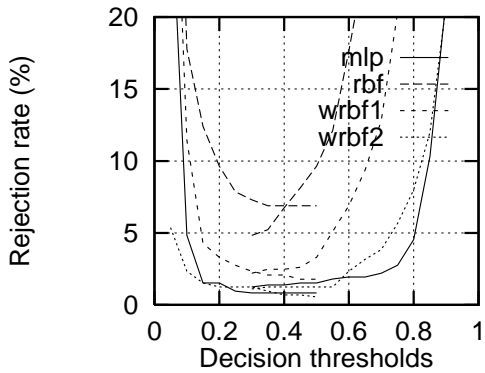


Figure 3: Rejection rate of an MLP, an RBF and two WRBFs versus the two decision threshold values σ_L and σ_H (left and right side of plot, respectively).

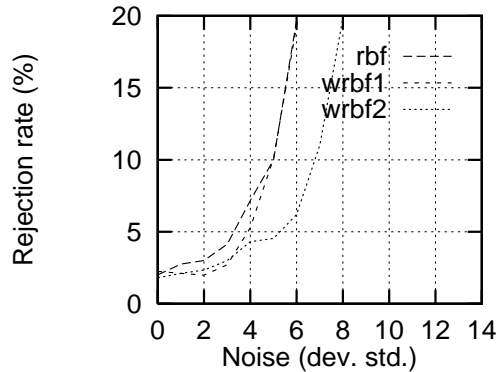


Figure 6: Noise sensitivity of an RBF and two WRBFs with 384 inputs (trained network).