New Radial Basis Function Neural Network Architecture for Pattern Classification: First Results

Humberto Sossa[1], Griselda Cortés[2], and Elizabeth Guevara[1]

[1] Instituto Politécnico Nacional-CIC
Miguel Othón de Mendizábal S/N, Gustavo A. Madero, México, D.F.C.P. 07738
[2] Tecnológico de Estudios Superiores de Ecatepec
Av. Tecnológico S/N, C.P. 55210. Col. Valle de Anáhuac
Ecatepec de Morelos, Estado de México
hsossa@cic.ipn.mx, griselda_cortes@tese.edu-mx,
eliza_uas@yahoo.com.mx

Abstract. This paper presents the initial results concerning a new Radial Basis Function Artificial Neural Network (RBFNN) architecture for pattern classification. Performance of the new architecture is demonstrated with different data sets. Its efficiency is also compared with different classification methods reported in literature: Multilayer Perceptron, Standard Radial Basis Neural Networks, KNN and Minimum Distance classifiers, showing a much better performance. Results are only given for problems using two features.

1 Introduction

Pattern classification is a main problem in pattern recognition. It consists on assigning a given pattern $X = [x_1, x_2, \dots, x_n]^T$ to one of p classes $C^k, k = 1,2, \dots, p$ by means of a mathematical tool called classifier. Examples of patterns that need to be classified are images, sounds, odours, and so on. The term pattern is used in many references, for example in [1], [2]. Multilayer and Radial Basis Function Neural Networks are two of the most often used tools for pattern classification, being used for classification and regression mostly in non-linearly separable situations [3], [4], [5] and for making predictions [6] and [7]. They are considered as machine learning techniques also providing discriminant function measurement vectors [8]. RBFNN are one of the most known techniques to define separating functions for solving classification problems in multi-dimensional spaces [9].

Usually, training of a RBFNN involves three steps: 1) finding the number of centroids needed to define the clusters to group the training samples, 2) specifying the clusters widths, and 3) adjusting the weights of output neuron of the network. The architecture of a RBFNN consists of three layers: an input layer, a hidden layer which uses radially symmetric functions such as a Gaussian kernels [10]; at this stage unsupervised learning (K-means algorithm) is used [1], [2], [10], [11] to define the hidden neurons, and an output layer with a linear node (case of two classes) that allows

E. Bayro-Corrochano and E. Hancock (Eds.): CIARP 2014, LNCS 8827, pp. 706–713, 2014.
© Springer International Publishing Switzerland 2014

calculating the weighted sum of the outputs from the hidden layer [6], [10] to compute the index class for an input pattern. One disadvantage when training any machine learning method is that when using a large and possibly noisy data could result in a poor learning. Also, overlapping of patterns classes will tend to severally affect accuracy of training and generalization [6] and [10].

To alleviate these problems, in this paper we present the first results concerning a new architecture for a RBFNN. One main advantage of our proposal is that the radial basis functions are completely automatically generated with no human intervention. The proposed training method ensures precise clustering of the training patterns, drastically reducing the training error and also increasing the generalization capability of the trained NN. Although the proposal is applicable to the n-dimensional case, in this paper only the two dimensional case is discussed. It is worth mentioning that in the general case training of the proposed RBFNN is 2^n.

The rest of the paper is organized as follows: in Section 2 the new architecture of RBFNN is presented. Section 3 is focused on describing the adopted training algorithm. Illustrative results of the functioning of the proposal and a comparison of its performance with other reported methods is provided in Section 4. Finally, Section 5 presents conclusions and directions for further work.

2 Proposed Architecture

As shown in Fig. 1, the proposed RBFNN architecture is composed of three layers and a decision function. As we will soon see, this new architecture and its associated training algorithm will allow efficiently classifying n-dimensional patterns to one of p classes, $C^k, k = 1,2, \dots, p,$.

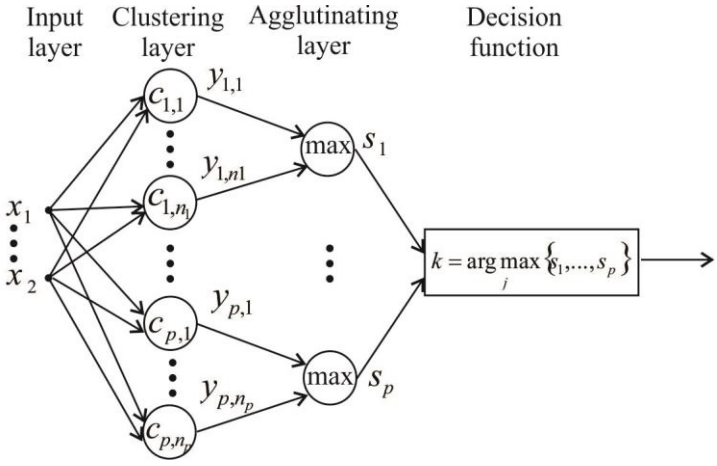


Fig. 1. The proposed architecture of a RBFNN for pattern classification

Input Layer: This layer receives n-dimensional vectors of the form:

$$X = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R} \tag{1}$$

and sends them directly to all the nodes of the next layer, the clustering layer.

Clustering Layer: This layer has two functions. During training it is used to partition the set of training vectors: $X^1, X^2, \ldots, X^M$ (some of them belonging to class C^1, others to class C^2, and so on) into a set of groups in such a way that each group contains only patterns belonging to the same class. As we will see, each group has the form of a hyper-rectangle; in the 2-bidimensional the form of a rectangle. The method adopted to get this partition was recently introduced in [12]. During classification this layer is used to computing the distances of an input pattern to each of the rectangles.

Agglutinating Layer: This layer contains p neurons. Neuron k has the function to select from all the hyper-rectangles of class k the rectangle to which the input pattern is closer. For this, neuron k performs the following operation:

$$s_k = \bigvee_{j=1}^{n_k} \{y_{k,1}, y_{k,2}, \ldots, y_{k,n_k}\} \tag{2}$$

In this case the symbol $\bigvee$ stands for the maximum of a set of numbers.

Decision Function: As can be seen from Fig. 1, this decision function takes the p outputs from the agglutinating layer and computes the following operation:

$$k = \underset{j}{\operatorname{argmax}}\{s_1, s_2, \ldots, s_p\} \tag{3}$$

Variable k will contain the index of the class to which an unknown pattern X should be classified after training.

3 Operation of the New Neural Network

Operation of the proposed neural network architecture is performed into two stages: training and testing. Each of these two stages is next explained in detail.

Training: Given a training set $\{X^1, X^2, \ldots, X^M\}$ and its corresponding set of labels $k = 1,2, \ldots, p$ of the classes to which the M vectors belong:

1. Cluster all M patterns by means of the algorithm introduced in [12]. In two dimensions, this step gives as a result a set of rectangles as depicted, for example, in Fig 2(b).
2. For each hyper-rectangle, take its center and assign to it a Gaussian kernel of the form (Fig. 2(c)):

$$G(X) = \exp\left(-\frac{\|X-\mu_j\|^2}{2\sigma_j^2}\right) \tag{4}$$

Spread σ_j^2 of the Gaussian is defined in terms of the dimensions of the corresponding box (Fig. 2(c)). If d is the dimension of the box, then $\sigma_j^2 = d$.

3. Take the n_k outputs of the k class and connect them directly to a max neuron as specified by equation (2). For an example refer to Fig. 2(d).
4. Finally, take the p outputs of the agglutinating layer and connect them directly to the function that performs the decision operation given by equation (3). For an example, refer to Fig. 2(e).

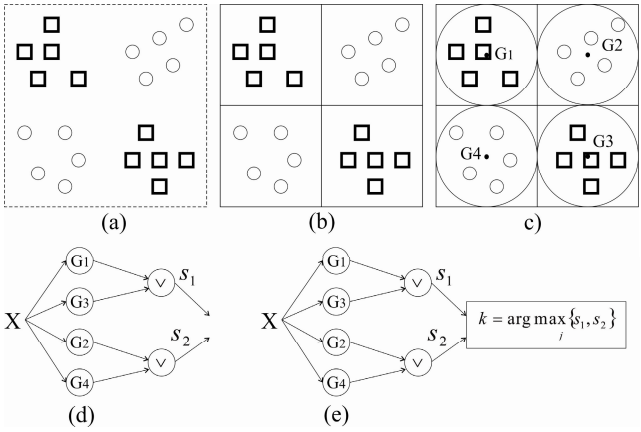


Fig. 2. (a) Set of patterns belonging to two classes; (b) Boxes generated by the first step of the training algorithm; (c) Centres and spreads of corresponding Gaussians; (d) Generation of the second and third layers of the Neural Network; (e) Generation of the output function.

Testing: Given a pattern X to be assigned to one p given classes $C^k, k = 1,2,\ldots,p$:

1. Present the pattern to the input of the already trained neural network.
2. Compute all the values y_{k,n_k} at the outputs of the neurons of the clustering layer using equation (4).
3. Compute all the p values $s_k, k = 1,2,\ldots,p$ at the outputs of the agglutinating layer using equation (2).
4. Finally, compute the value at the output of the decision function by using equation (3).

The integer value obtained at this stage is the class to which pattern X is assigned.

4 Design Example

To better appreciate the designing of the proposed neural network, let us take the set of patterns belonging to two classes C^1 and C^2 shown in Fig. 2(a). We can see that these patterns are non-linearly separable which is known to represent a challenge for many classifiers. By applying the first step of the training procedure described in the previous section we get the set of four boxes shown in Fig. 2(b).

Following step two of the training procedure, we obtain the four Gaussian kernels: $y_{1,1} = G(\mu_1, \sigma)$, $y_{1,2} = G(\mu_2, \sigma)$, $y_{2,1} = G(\mu_3, \sigma)$ and $y_{2,2} = G(\mu_4, \sigma)$. For this example, the sigma σ for the four Gaussians is the same due to all their boxes have the same size (Fig. 2(c)).

Now by applying the third step of the training procedure, we connect Gaussians 1 and 3 to first max neuron of agglutinating layer as shown in Fig. 2(d) and Gaussians 2 and 4 to second max neuron of this same layer as shown again in Fig. 2(d). Finally, we connect the outputs of the two neurons of the agglutinating layer to the decision neuron as shown in Fig. 2(e).

5 Results and Comparison

In this section we test the efficiency of the proposed architecture. We also compare its performance with several known classification paradigms. For this we use the six 2-dimensional sets of patterns shown in Figs. 3(a), 3(c), 3(e), 3(g), 3(i), and 3(k).

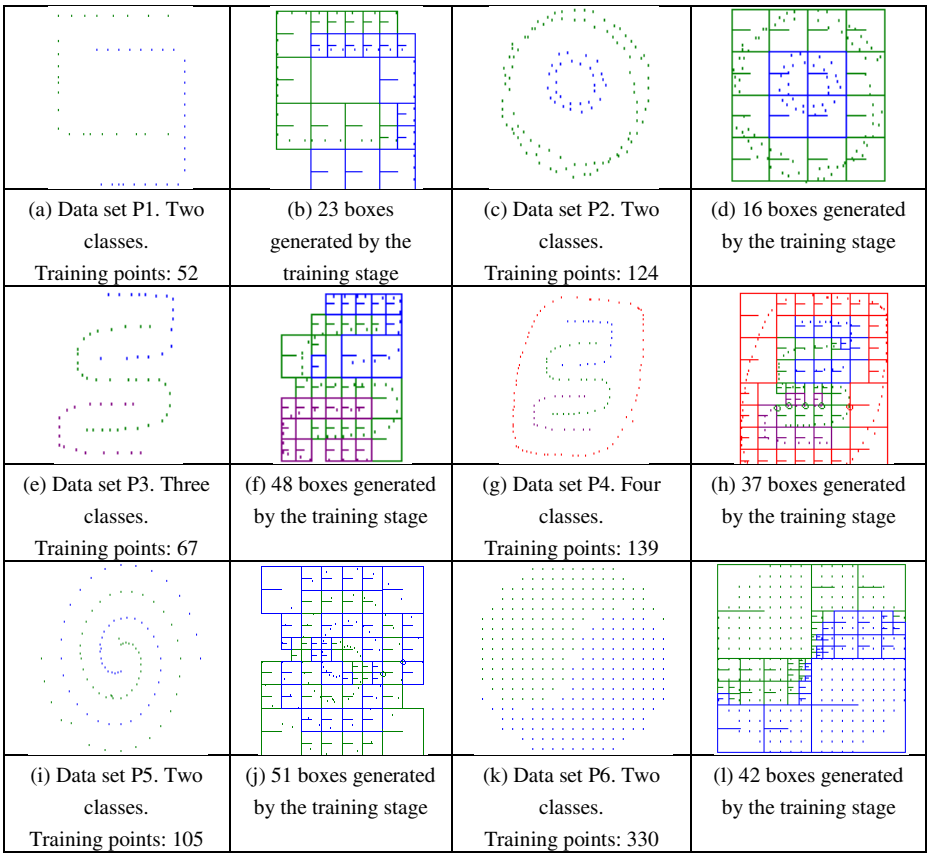
Figures 3(b), 3(d), 3(f), 3(h), 3(j), and 3(l) shows the corresponding rectangles for each of the six problems generated by the first step of the training procedure.
Table 1 shows the efficiency of the proposed RBFNN architecture compared with the classical RFFNN architecture.

The first column of this table depicts the name of each of the problems and the number of points used for testing. Columns 3 to 12 show the classification errors with six arrangements of the classical RBFNN with 10, 20, 40, 60, 80, and 100 clusters.

As can be seen from columns 3, 5, 7, 9, 11, and 13 the training error tends to decrease as the number of clusters increases. However from columns 4, 6, 8, 10, 12, and 14 we appreciate that the average testing error is more or less the same and it is above 10%. From columns 16 and 17 we see that the training and testing errors produced by our proposal are very small, 1.02% and 1.9%, respectively. So far, our proposal performs much better than the standard RBFNN.

Table 2 shows the efficiency of the proposal compared with other three classical classifiers: the minimum distance classifier, the KNN classifier and the well-used MLP neural network. From this table we see that the minimum distance classifier has the worst performance. This is a normal result considering the nature of the problems involved in the analysis.

The MLP classifier performs much better than the minimum distance classifier. This is also a normal result. On the other hand, we see that the KNN classifier exhibits the best performance among the three, however our proposal provides the best performance.



(a) Data set P1. Two classes. Training points: 52 | (b) 23 boxes generated by the training stage | (c) Data set P2. Two classes. Training points: 124 | (d) 16 boxes generated by the training stage

(e) Data set P3. Three classes. Training points: 67 | (f) 48 boxes generated by the training stage | (g) Data set P4. Four classes. Training points: 139 | (h) 37 boxes generated by the training stage

(i) Data set P5. Two classes. Training points: 105 | (j) 51 boxes generated by the training stage | (k) Data set P6. Two classes. Training points: 330 | (l) 42 boxes generated by the training stage

Fig. 3. The six data sets used to test and compare the performance of the proposal

Table 1. Effectiveness of the proposed RBFNN compared with the standard RBFNN architecture. Tt stands for training error, Ts for testing error, while k is the number of rectangles generated for each problem by the training procedure.

Problem and		Number of clusters														
Number of		10		20		40		60		80		100		New NN		
Testing points		% of error														
		Tr	Ts	Tr	Ts	Tr	Ts	Tr	Ts	Tr	Ts	Tr	Ts	k	Tr	Ts
P1	189	0	28	0	21	0	25	0	25	0	25	0	25	23	0	2.6
P2	236	0	8.9	0	8.5	0	8.9	0	8.1	0	8.5	0	8.9	16	0	0.4
P3	271	0	22	0	17	0	14	0	14	0	14	0	14	37	2.99	0.7
P4	496	0	17	0	19	0	20	0	19	0	18	0	18	51	2.16	3.8
P5	500	16	18	1.9	16	3.8	15	1.9	8	1.9	8	1.9	8	48	0.95	3.8
P6	816	0	0.5	0	2.5	0	2.3	0	0.9	0	3.6	0	1.5	42	0	0.3
Tot	Err	2.7	16	0.3	14	0.6	14	0.3	12	0.3	13	0.3	12		**1.02**	**1.9**

Table 2. Effectiveness of the proposed RBFNN compared with the minimum distance classifier, the KNN classifier and the MLP trained with back-propagation rule. Tr is the training error, Ts the testing error, *K* is the number of distances taken by the KNN classifier, *m* is the number of hidden layers of the MLP, and *k* is the number of boxes generated for each problem by the training procedure.

Problem and number of testing points		Minimum distance classifier		KNN classifier			MLP classifier			New NN		
		% error			% error			% error			% error	
		Tr	Ts	*K*	Tr	Ts	*m*	Tr	Ts	*k*	Tr	Ts
P1	189	21.2	20.6	4	0	12	2	17.3	25.4	23	0	2.6
P2	236	47.6	50.4	21	0	6.4	2	18,5	22	16	0	0.4
P3	271	26.9	27.3	1	0	5.5	2	23.9	25.1	37	2.99	0.7
P4	496	66.9	63.9	1	0	9.9	3	15.8	27.2	51	2.16	3.8
P5	500	48.6	43	29	0	0	2	45.7	44	48	0.95	3.8
P6	816	7.27	19	1	0	0.4	2	2.12	9.19	42	0	0.3
Total	Error	36.4	37.4		0	5.7		20.6	25.5		1.02	**1.9**

6 Conclusions and Further Work

This paper proposed a new RBF model for pattern classification. The proposal is applicable to problems with p classes and n features, although here only the results for problems with two features were presented.

The new RBFNN architecture is composed of three layers and uses the clustering procedure introduced in [12] as an essential step for its training.

One main advantage of the new RBFNN architecture is that is does not require that the user specifies the number of cluster into which the data have to be divided, the adopted method does this automatically.

Another major advantage of the proposed RBFNN architecture is that its training is always performed in reduced finite number of iterations.

From the experiments with six non-linearly separable problems we have seen that our proposal performs much better than the standard RBFNN architecture. We have also shown that it performs much better that other well-known classifiers: minimum distance classifier, KNN classifier and MLP classifiers.

Further work comprises: 1) testing the efficiency of the proposal with problems with more than two features and in more realistic situations, 2) comparing with other classifiers such as support vector machines, and 3) testing with other kernels different from Gaussians, among others.

Acknowledgements. Humberto Sossa would like to thank SIP-IPN and CONACYT under Grants 20140776 and 155014 for economical support to carry out this research. Griselda Cortés thanks Tecnológico de Estudios Superiores de Ecatepec for the support to pursue her doctoral studies.

References

1. Sanz, G.M., de la Cruz García, J.M.: Visión por computadora Imágenes digitales y aplicaciones, 2nd edn. Editorial Alfaomega (2008)
2. Cruz, P.P.: Inteligencia Artificial con Aplicaciones a la Ingeniería. Editorial Alfaomega (2010)
3. Ritter, G.X., Laurentiu, I., Urcid, G.: Morphological perceptron with dendritic structure. In: The 12th IEEE International Conference on Fuzzy System, pp. 1296–1301 (2003)
4. Ritter, G.X., Sussner, P.: Morphological perceptrons. Intelligent System and Semiotics, pp. 221–226 (1997)
5. Ritter, G.X., Urcid, G.: Lattice Algebra Approach to Single-Nueron Computation. IEEE Transactions on Neural Networks 14(2), 282–295 (2003)
6. Skomorokhov, A.: Radial Basis Function Networks in A+. In: Proceedings of the Conference on APL 2002, vol. 32(4), pp. 198–213 (2002)
7. Sug, H.: An Empirical Improvement of the Accuracy of RBF Networks. In: Proceedings of the Second International Conference on Interaction Sciences, ICIS 2009, pp. 708–712 (2009)
8. Bishop, C.M.: Neural Network for Pattern Reconigtion, 1st edn. Oxford University Press, New York (2008)
9. Liu, J., Lampinen, J.: A Differential Evolution Based Incremental Training Method for RBF Networks. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, GECCO 2005, pp. 881–888 (2005)
10. Pandya, A.S., Macy, R.B.: Pattern Recognition With Neural Networks in C++. IEEE Press (1995)
11. Principe, J.C., Euliano, N.R., Curt Lef, W.: Neural and Adaptive Systems Fundamentals Through Simulations. In: Library of Congress Cataloging-in-Publication Data (2000)
12. Sossa, H., Guevara, E.: Efficient Training for dendrite morphological neural networks. Neurocomputing 131, 132–142 (2014)