A preliminary study of the effect of feature selection in evolutionary RBFN design

M.D. Pérez-Godoy J.J. Aguilera F.J. Berlanga V.M. Rivas A.J. Rivera
Department of Computer Science, University of Jaen
{lperez,jjaguile,berlanga,vrivas,arivera}@ujaen.es

Abstract

In this paper, the effect of the inclusion of a feature selection stage previous to the RBFNs design is analyzed.
Two different RBFNs design algorithms have been used: a cooperative-competitive scheme, where each individual is a single neuron, and a Pittsburgh evolutionary scheme, where each individual is a complete network. On the other hand, six different feature selection algorithms (three filter and three wrapper) have been considered.
The experimentation shows the generalization ability of the obtained RBFNs (with and without applying feature selection). Furthermore the inclusion of an FS stage leads to less complicated network structure and thus increases the simplicity of the system and the efficiency in processing data.

Keywords: RBFNs, Feature Selection, Fuzzy Rule Base Systems.

1 Introduction

Radial Basis Function Networks (RBFNs) are one of the most important Artificial Neural Network paradigms in the machine learning field. They have been successfully used in many areas such as pattern classification, function approximation, and time series prediction, among others. RBFNs have interesting characteristics such as their simple topological structure and their high degree of interpretability. The functional equivalence between RBFNs and TSK Fuzzy Inference Systems (FISs) have been established in [8][18].

FISs, also known as Fuzzy Rule-Based Systems (FRBSs), are a popular computing framework based on the concepts of Fuzzy Set Theory, fuzzy IF-THEN rules, and fuzzy reasoning [9][22]. The basic structure of an FRBS consists of three conceptual components: a rule base, containing a set of fuzzy rules; a database, which defines the membership functions used in the fuzzy rules and a fuzzy reasoning method, which performs the inference procedure upon the rules and given facts to derive a reasonable output.

On the other hand, any inductive supervised learning algorithm has to face up to two difficulties when dealing with high-dimensional problems: firstly, it should be able to adequately deal with a very high search space (since an increase in the number of variables leads to an exponential grow in the fuzzy rules search space); secondly, the obtained system could turn to be too complex, being this one of the crucial aspect in the interpretability of FRBS and, consequently, RBFNs.

Both difficulties can be solved by using a preprocessing algorithm such as a feature selection (FS) process. An FS process can be defined as a preprocessing algorithm that searches for a subset of the complete set of variables in order to obtain simpler systems with a greater precision in the forecasting.

Genetic Algorithms (GAs) [5] for instance, have been successfully applied to carry out FS processes [1][3][14][19][23].

In this paper, we study and analyze the effect of FS procedures in the evolutionary design of RBFNs for high-dimensional classification problems. Different RBFNs design methods for classification problems have been set out in the specialized bibliography, some of them using evolutionary algorithms (see [2], [11], [13] among others). FS algorithms have also been applied in order to reduce the complexity of the obtained RBFNs [6]. In this paper a feature ranking algorithm based on a separability correlation measure is considered and the RBFN design method is iteratively applied with a increasing set of features in order to select the best RBFN generalization results. In our paper, six different FS algorithms have been used to pre-process a set of classification databases, and, after this, generalization abilities of the obtained RBFNs, with and without applying FS, are compared.

The contribution is organized as follows: Section 2 introduces some preliminary concepts; Section 3 describes the six FS methods and the two RBFN evolutionary learning methods used in this work; the experimental study carried out is shown in Section 4; finally, conclusions are presented in Section 5.

2 Preliminary Concepts

2.1 Feature selection

Feature Selection (FS) techniques select the best subset of features from the original set. The features that are important to maintain the concepts in the original data are selected from the entire feature set.

Besides the search algorithm, how to determine the importance of an individual feature or a feature subset is the key to any FS technique. Depending on this quality function, the FS algorithms can be classified as [10],[12]: *filter* models, that use evaluation measures based on separability of classes; and *wrapper* models, that use an estimation of the precision in the classification process (designing a classification system from the selected variables).

GAs have been developed for feature selection by using both filter and wrapper approaches [1][3][7][14][19][23].

2.2 Classification with Radial Basis Function Networks

An RBFN is a feed-forward neural network with three layers: an input layer with n nodes, a hidden layer with m neurons or RBFs, and an output layer with one or several nodes. The m neurons of the hidden layer are activated by a radially-symmetric basis function, $\phi_i : R^n \rightarrow R$, which can be defined in several ways, being the Gaussian function the most widely used: $\phi_i(\vec{x}) = \phi_i(e^{-(\|\vec{x}-\vec{c_i}\|/d_i)^2})$, where $\vec{c_i} \in R^n$ is the center of basis function ϕ_i, $d_i \in R$ is the width (radius), and $\|\|$ is typically the Euclidean norm on R^n. This expression is the one used in this paper as Radial Basis Function (RBF). The output nodes implement the following function:

$$f_i(\vec{x}) = \sum_{i=1}^{m} w_{ij}\phi_i(\vec{x}) \tag{1}$$

In a classification environment, the RBFN has to perform a mapping from an input space X^n to a finite set of classes Y with k classes. For this, a typical training set S is defined as:

$$\begin{aligned} S = \{(\vec{x}_u, y_u) | x_u \in X^n, \\ y \in Y, u = 1, \ldots, p\} \end{aligned} \tag{2}$$

where $\vec{x}_u$ is the feature vector and y_u is the class it belongs to. Usually, in the classification scenario, the number of outputs of the RBFN corresponds to the number of classes (k). In order to train the network, the class membership y_u is encoded into a binary vector through the relation $\vec{z}_u^i = 1$ iff $y_u = i$, and $\vec{z}_u^i = 0$ otherwise. The output class of the network will be the output node with maximum activation.

3 Feature Selection and Evolutionary RBFNs considered algorithms

3.1 Feature Selection

In this paper we have used six FS algorithms [12]:

- The FS-LVF algorithm. It is a classic, non-evolutionary filter stochastic method, which uses the inconsistency measure as quality function, and Las Vegas as search procedure.

- The FS-Focus algorithm. It is a classic, non-evolutionary filter method, which uses the inconsistency measure as quality function and considers all the combination among N features, starting from an empty subset.

- The FS-Forward and FS-Backward algorithms. They are two classic, non-evolutionary wrapper greedy methods. As quality function they use the precision estimation provided by the K-NN algorithm [4] using the selected features.

- The FS-GGA algorithm. It is a generational GA filter with binary representation, that uses k-tournament selection, one-point crossover, and one-point binary mutation. The fitness function is defined as $((1-\lambda) * inconsistency\_rate + \lambda * selected\_features)$, where λ and $inconsistency\_rate$ are given as input parameters.

- The FS-SSGA algorithm. It is a steady state GA wrapper with integer codification. It uses the one-point crossover operator, random integer mutation, and as fitness function, the precision estimation provided by the K-NN algorithm [4] using the selected variables represented in the chromosome.

3.2 CO^{2}RBFN: a cooperative-competitive hybrid algorithm for RBFNs design

In the specialized bibliography few cooperative-competitive procedures have been implemented up to now [17],[20],[21] for RBFNs design. The difficulty of this kind of methods lies in the credit assignment strategy which must promote competition among similar RBFs and cooperation among different ones at the same time.

A cooperative-competitive design of RBFNs (CO^{2}RBFN) for solve classification problems was proposed in [15]. In this approach, each individual of the population represents a basis function, thus the entire population represents the final solution, and the individuals cooperate towards a definitive solution. However, they also compete for survival, since if their performance is poor they will be eliminated. This cooperation-competition scenario reinforces the local operation (neurons with local response) and the interpretability of this kind of network, and establishes an important design guideline in this algorithm.
In order to measure the credit assignment of an individual, three factors have been proposed to evaluate the role of the i-th RBF in the network:

Contribution, a_i: is determined by considering the weight, w_i, and the number of patterns of the training set inside its width.

Error, e_i: is obtained as the proportion of wrongly classified patterns inside the i-th RBF radius.

Overlapping, o_i: the overlapping of the i-th RBF is calculated as the sum of the overlapping between the i-th RBF and the others.

In this proposal four operators have been proposed:

Remove: it substitutes bad individuals by new ones located in the center of the largest zones wrongly classified outside of any RBF width or by new ones randomly created.

Mutation with Information: it modifies the RBFs width and coordinates of their centers. The objective of the width modification is that most of the patterns belonging to the RBF class will be inside the RBF radius. The center is varied in order to approximate it to the average of the patterns which belong to the RBF class and are inside its radius.

Mutation without Information: it modifies the width and some coordinates of the center by means of a random quantity.

Null: no operator is applied to the RBF when this operator is selected .

These operators will be applied to the whole population of RBFs. In order to decide the operators' application probability over a certain RBF, the algorithm uses a Mamdani FRBS whose inputs are the parameters which determine the credit assignment to each RBF and the outputs are the probability of applying the operators. To design the set of rules, we take into account the fact that an RBF is worse if its contribution (a_i) is low, its error (e_i) is high, and its overlapping (o_i) is also high. On the other hand, an RBF is better when its contribution is high, its error is low and its overlapping is also low. Therefore as the associated RBF becomes worse, the elimination probability increases. However, as the associated basis function improves the null operator application probability increases.
The main steps of CO^{2}RBFN are shown in figure 1. In step 2, the well-known LMS algorithm is used.

```
1.  Initialize RBFN
2.  Train RBFN
3.  Evaluate RBFs
4.  Apply operators to RBFs
5.  Substitute the RBFs that were
    eliminated
6.  Select the best RBFs
7.  If the stop-condition is not
    verified go to step 2
```

Figure 1: Main steps of CO^{2}RBFN.

3.3 The EvRBF algorithm

EvRBF [16] is a steady state evolutionary algorithm that includes elitism. It follows the Pittsburgh scheme, in which each individual is a full RBFN whose size can vary, while population size remains equal.

EvRBF codifies in a straightforward way the parameters that define each RBFN, using an object-oriented paradigm; for this reason, it includes operators for recombination and mutation that directly deal with the neurons, centers and widths themselves.

Recombination operators (X_FIX and X_MULTI) interchange information between individuals, trying to find the building blocks of the solution.

X_FIX: replaces a sequence of hidden neurons in RBFN R_1 by a sequence of the same size taken from RBFN R_2.

X_MULTI: replaces with probability p_{x_multi} every hidden neuron in RBFN R_1 by a randomly chosen neuron coming from net R_2.

Mutation operators (centers and width modification: C_RANDOM and R_RANDOM) use randomness to increase diversity generating new individuals so that local minima can be avoided. Furthermore, EvRBF tries to determine the correct size of the hidden layer using the operators ADDER and DELETER to create and remove neurons, respectively.

C_RANDOM: modifies RBF centers, allowing to explore the input space. It swaps the current value, c_i, for a random value following an uniform probability function.

R_RANDOM: modifies the widths using an uniform probability function.

ADDER: adds a single neuron with random center and width, using an uniform probability function. Selection pressure will reward or penalize nets containing the new neuron according to its effect in the fitness function.

DELETER: randomly removes neurons from the RBFN to which is applied.

The exact number of neurons affected by these operators (except ADDER) is determined by their internal application probabilities.

EvRBF incorporates tournament selection for reproduction. The skeleton of EvRBF, showed in figure 2, is commonly used in evolutionary algorithms. The *fitness function* measures the generalization capability of each individual as the percentage of samples it correctly classifies. When comparing two individuals, if and only if both two individuals have exactly the same error rate, the one with less neurons is said to be better. As in CO^2RBFN the LMS algorithm is used in order to train individuals.

```
1.  Create, train, evaluate and set
    fitness of first generation.
2.  Until stop condition is reached
    (a) Select and copy individuals
        from current population.
    (b) Modify, train, and set
        fitness of the copies.
    (c) Replace worst individuals by
        the copies.
3.  Train last generation with
    training and validation data
4.  Use test data set to obtain the
    generalization ability of each
    individual.
```

Figure 2: General skeleton of EvRBF.

4 Experimentation and analysis of results

For the experimentation carried out, four databases[1] have been used: *Ionosphere* (351 instances, 34 features, 2 classes);*Sonar* (208 instances, 60 features, 2 classes); *Vehicle* (846 instances, 18 variables, 4 classes) and *Wdbc* (570 instances, 30 features, 2 classes).

The 10-fold cross-validation (*10-fcv*) method has been used to estimate the accuracy of FS and RBFNs design algorithms and every algorithm has been run five times.

The preprocess methods have been executed as follows: in all necessary cases, the number of nearest neighbours used to estimate the accuracy is 1. Particularly, for the FS-LVF algorithm the number of loops executed before obtain the final solution has been $77{*}n$ and the inconsistency rate has been set to 0 (too for the FS-Focus algorithm). For both genetic algorithms the number of evaluations performed has been established to 5000 and the population size to 100. For the FS-GGA algorithm the parameter λ used in the fitness function that weighs up the precision rate and the feature reduction has been 0.1, the crossover probability: 0.6, the mutation probability: 0.01 and the number of best features selected before applying the random tournament to selected one of them 5. Finally, for the FS-SSGA, the number of features to be selected was established for each data set as the same number obtained by the LVF method.

CO^2RBFN has been executed using its typical value parameters [15]: 200 generations and 8 individual per generation. The random quantities used in mutation operators are set between 0.05 and 0.25 of the widths. As in previous works [16], EvRBF has been executed using 100 generations and 100 individuals per generation; the probabilities of operators were 0.4 for recombination (2 operators), and 0.05 for mutation (4 operators); internal application probabilities (for mutator that required

[1]Obtained from the UCI Repository `http://www.ics.uci.edu/~mlearn/MLRepository.html`

Table 1: Experimental Result with Ionosphere database

Preprocessing	**CO^{2}RBFN**			**EvRBFN**		
	Nodes	**Param.**	**Test (%)**	**Nodes**	**Param.**	**Test (%)**
none	8	296	93.25	10	370	96.65
FS-GGA	8	80	83.39	12	120	93.55
FS-LFV	8	96	87.95	10	120	93.56
FS-SSGA	8	96	89.98	9	108	95.16
FS-BackwardLVO	8	256	91.61	14	448	96.34
FS-Focus	8	56	88.57	15	105	93.70
FS-ForwardLVO	8	48	92.49	9	54	95.05

Table 2: Experimental Result with Sonar database

Preprocessing	**CO^{2}RBFN**			**EvRBFN**		
	Nodes	**Param.**	**Test (%)**	**Nodes**	**Param.**	**Test (%)**
none	8	504	75.27	6	378	70.20
FS-GGA	8	136	70.47	8	136	80.49
FS-LFV	8	160	73.42	8	160	79.30
FS-SSGA	8	160	72.45	9	180	81.40
FS-BackwardLVO	8	336	73.68	10	420	81.34
FS-Focus	8	56	75.80	6	42	82.94
FS-ForwardLVO	8	72	74.51	4	36	80.90

them) were set to 0.5; tournament size was set to 2, and 30% of individuals were replaced in every new generation.

Tables 1, 2, 3, and 4 show the results obtained by CO^{2}RBFN and EvRBFN algorithms, with and without a preprocess FS method. An analysis of the results shows that:

- The evolutive algorithms for the RBFNs design have obtained good results with high-dimensional problems, and reach precise solutions without using a FS method.
- The coevolutive proposal, CO^{2}RBFN, obtains good results without FS.
- The evolutive proposal, EvRBFN, sometimes improves with preprocessing since its Pittsburgh evolutionary scheme is more sensitive to increases in the dimensionality; thus, it yields better results when FS is used to reduce the dimensionality.
- The best results are obtained using the FS-SSGA and the FS-ForwardLVO algorithms due to their fitness function is based on distances, this is, in the same way RBFNs operate.
- In all the databases the inclusion of a FS stage before the evolutionary RBFN design approaches reduces the execution time in the design process, leads to less complicated network structure and thus increases efficiency in processing data.
- Even more, although the evolutionary proposals in most of the databases considered do not require a preprocessing stage to address high-dimensional problems, it is needed for obtaining a simpler fuzzy system, with fewer conditions in the rule antecedents, maintaining good classification rates.

5 Conclusions

In this paper, the behaviour of RBFNs design methods including a previous preprocessing stage (with six different feature selection algorithms) have been analyzed.

Three filter and three wrapper feature selection algorithms have been applied, and two types of RBFNs design algorithms have been

Table 3: Experimental Result with Vehicle database

Preprocessing	**CO^{2}RBFN**			**EvRBFN**		
	Nodes	**Param.**	**Test (%)**	**Nodes**	**Param.**	**Test (%)**
none	8	184	58.35	44	1012	57.48
FS-GGA	8	56	51.96	50	600	59.72
FS-LFV	8	104	53.36	55	715	60.29
FS-SSGA	8	104	58.56	47	611	61.33
FS-BackwardLVO	8	144	60,09	56	1008	68.08
FS-Focus	8	96	50.26	58	696	61.92
FS-ForwardLVO	8	96	60.85	54	648	65.11

Table 4: Experimental Result with Wdbc database

Preprocessing	**CO^{2}RBFN**			**EvRBFN**		
	Nodes	**Param.**	**Test (%)**	**Nodes**	**Param.**	**Test (%)**
none	8	264	96.27	7	231	95.51
FS-GGA	8	80	95.85	12	120	96.42
FS-LFV	8	88	94.51	8	88	92.76
FS-SSGA	8	88	95.53	6	66	94.97
FS-BackwardLVO	8	240	96.37	8	240	95.47
FS-Focus	8	72	94.76	10	90	93.60
FS-ForwardLVO	8	104	96.48	9	117	95.40

considered: a cooperative-competitive scheme where each individual is a neuron, and a Pittsburgh evolutionary scheme where individuals are networks.

The experimentation carried out have shown that the algorithms used to design RBFNs deal well with high dimensionality and the results obtained by these algorithms with or without preprocessing stage are similar. Nevertheless, the FRBSs extracted from the solution reached by the RBFN design algorithms is simpler when a feature selection stage is applied. The use of a FS wrapper approach and a quality measure based on distance (as the k-nearest neighbour rule) is important as preprocessing stage in the RBFN design.

Acknowledgements

This work has been partially supported by the CICYT Spanish TIN2005-04386-C05-03.

References

[1] J. Bala; K. De Jong; J. Huang; H. Vafaie; H. Wechsler (1997). Using learning to facilitate the evolution of features for recognizing visual concepts. *Evolutionary Computation*, Vol. 4, n. 3, pages 297-311

[2] O. Buchtala; M. Klimek; B. Sick (2005). Evolutionary optimization of radial basis function classifiers for data mining applications. *IEEE Transactions on Systems, Man and Cybernetics*, Part B, Vol. 35, n. 5, pages 928-947.

[3] J. Casillas; O. Cordón; M.J. Del Jesus; F. Herrera (2001). Genetic feature selection in a fuzzy rule-based classification system learning process for high-dimensional problems. *Information Sciences*, Vol. 136, pages 135-157

[4] T.M. Cover; P.E. Hart (1967). Nearest neighbor pattern classification. *Trans IEEE Inform Theory* Vol. 13, pages 21-7.

[5] D.E. Goldberg (1989). Genetic Algorithms in Search, Optimisation and Machine Learning. *Addison-Wesley* 1989.

[6] X. Fu; L. Wang (2003). Data Dimensionality Reduction With Application to Simplifying RBF Network Structure and Improving Classification Performance. *IEEE Trans. on Systems, Man,*

and Cybernetics - Part B: Cybernetics, Vol. 33, n. 3, pages 399-409.

[7] A. González; R. Pérez (2001). Selection of relevant features in a fuzzy genetic learning algorithm. *IEEE Transactions on Systems, Man and Cybernetics* Part B, Vol. 31, n. 3, pages 417-425

[8] K.J. Hunt; R. Haas; R. Murray-Smith. (1996). Extending the Functional Equivalence of Radial Basis Function Networks and Fuzzy Inference Systems. *IEEE Transaction on Neural Networks*, Vol. 7, n. 3, pages 776-781.

[9] G.J. Klir; B. Yuan (1995). Fuzzy Sets and Fuzzy Logic. *Prentice-Hall*, Englewood Cliffs, NJ.

[10] R. Kohavi; G.H. John (1997). Wrappers for Feature Subset Selection. *Artificial Intelligence*, Vol. 97, pages 273-324

[11] E. Lacerda; A. Carvalho; A. Braga; T. Ludermir (2005). Evolutionary Radial Functions for Credit Assessment. *Applied Intelligence 22. Springer Netherlands*, pages 167-181.

[12] H. Liu; H. Motoda (1998). Feature Selection for Knowledge Discovery and Data Mining *Ed. Kluwer Academic Publishers.*

[13] R. Neruda; P. Kudov (2005). Learning methods for radial basis function networks. *Future Generation Computer Systems*, Vol. 21, issue 7, pages 1131-1142.

[14] I.S. Oh; J.S. Lee; B.R. Moon (2004). Hybrid Genetic Algorithms for Feature Selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 26, n. 11, pages 1424-1437.

[15] M.D. Pérez-Godoy; A.J. Rivera; M.J. del Jesus; I. Rojas (2007). CoEvRBFN: an approach to solving the classification problem with a hybrid cooperative-coevolutive algorithm. *Proceedings of the International Work-Conference on Artificial Neural Networks (IWANN)*, pages 324-332, San Sebastián, Spain.

[16] V.M. Rivas; J.J. Merelo; P.A. Castillo; M.G. Arenas; J.G. Castellanos (2004). Evolving RBF neural networks for time-series forecasting with EvRBF. *Information Sciences*, Vol. 165, n. 3-4, pages 207-220.

[17] A. J. Rivera; I. Rojas; J. Ortega; M.J. del Jesus (2006). A new hybrid methodology for cooperative-coevolutionary optimization of radial basis function networks. Soft Computing. ISSN 1432-7643. D.O.I: http://dx.doi.org/10.1007/s00500-006-0128-9.

[18] J.S. Roger Jang; C.T. Sun (1993). Funtional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems. IEEE Transaction on Neural Networks, Vol 4, n. 1, pages 156-159.

[19] W. Siedlecki; J. Sklansky (1989). A note on genetic algorithm for large-scale feature selection. *Pattern Recognition Letters*, Vol. 10, pages 335-347.

[20] A. Topchy; O. Lebedko; V. Miagkikh; N. Kasabov (1998). Adaptive training of radial basis function networks based on cooperative evolution and evolutionary programming. Prog. in connectionist-based information syst. N. Kasabov et al (eds), Springer, pages 253-258.

[21] B. Whitehead; T. Choate (1996). Cooperative-competitive genetic evolution of Radial Basis Function centers and widths for time series prediction. IEEE Trans. on Neural Networks, Vol.7, n. 4, July, pages 869-880.

[22] L.A. Zadeh (1965). Fuzzy sets. *Information and Control*, Vol.8, pages 338-353.

[23] P. Zhang; B. Verma; K. Kumar (2005). Neural vs. statistical classifier in conjunction with genetic algorithm based feature selection. *Pattern Recognition Letters*, Vol. 26, n. 7, pages 909-919.