EvRBF: Evolving RBF Neural Networks for classification problems

Victor M. Rivas, Maribel G. Arenas
University of Jaen
Department of Computer Science
Campus Las Lagunillas S/N, 23071, Jaen
SPAIN

Juan J. Merelo, Alberto Prieto
University of Granada
Department of Architecture
and Computers Technology
ETSII de Granada
SPAIN

Abstract: This paper shows the latest results of the EvRBF algorithm applied to classification problems. EvRBF is an evolutionary algorithm intended to automatically design Radial Basis Functions Neural Networks (RBFNN), being its main advantage that establishes the whole set of parameters of the nets.

Key–Words: Radial basis function neural networks, evolutionary algorithms, evolutionary operators,EvRBF, classification, evolutionary algorithms representation.

1 Introduction

Radial Basis Function Neural Networks (RBFNN) [9] are two-layer, fully-connected, feed-forward networks, in which hidden neuron activation functions are Radial Basis Functions (RBF), usually Gaussian.

RBFNN output is given by eq. 1.

$$s_j(\vec{x_k}) = \lambda_{0j} + \sum_{i=1}^{p'} \lambda_{ij}\phi_i(\vec{x}, \vec{c_i}, \vec{r_i}) \qquad (1)$$

where $k = 1..p, j = 1..n', s_j \in R, \vec{x_k} \in R^n$, and ϕ_i is the RBF assigned to hidden neuron i; λ_{0j} is a bias term; λ_{ij} represents the weight between hidden neuron i and output neuron j; $\vec{c_i}$ and $\vec{r_i}$ are called, respectively, the *center* and *radii* (or *widths*) of the RBF; n and n' are the input and output space dimensions, respectively; p' is the number of hidden neurons, and p is the number of patterns to which s_j is going to be applied.

RBFNN's main advantage is that optimal biases and weights (i.e, λ_{0j} and λ_{ij}) can be efficiently computed for a certain set of desired output, once the number of hidden neurons, centers and radii have been set.

This paper shows the latest results obtained by EvRBF, an evolutionary algborithm designed to dynamically build the optimal RBFNN that solves a classification problem. Unlike many other methods [7], EvRBF is able to set all the parameters that configure the net. It includes RBFNN-specific operators that modify the RBFNN structure, and does not limit the maximum number of neurons. Despite being an evolutionary algorithm, is quite fast algorithm, includes straightforward operators and easy fitness computation, and is guided by the nets generalization ability in its searching task.

The rest of the paper is organized as follows: Section 2 reviews methods in literature intended to generate RBFNN. Section 3 describes EvRBF algorithm, paying a special attention to the operators. Section 4 shows the experiments carried out to test EvRBF in classification problems, and compares to other methods. Finally, section 5 describes conclusions and future research lines.

2 State of the art

Harpham et al. [7] reviewed some of the most well known methods that apply evolutionary algorithms to RBFNN design. They concluded that, in general, methods tend to concentrate in only one aspect when designing RBFNN. Nevertheless, there also exist methods intended to optimize the whole net, such as [11] for Learning Vector Quantization (LVQ) nets or [5] for multilayer perceptrons.

Chronologically, first papers [3, 16] used binary codification and were constricted by the number of hidden neurons, that had to be set a priori. While Carse [3] works with a population of nets, Whitehead [16] works with a population of neurons (thus, only one RBFNN was build and evaluated) that compete and cooperate to find the optimal net.

Subsequent papers [2] presented algorithms based on real number representation. Newly, the imposition of a limit to the number of neurons, optimization of only the centers for RBF, and a badly defined penalization term for the number of times an individual can

reproduce, are the main drawbacks of this algorithm.

Recent applications try to overcome the disadvantages of the preceding methods. Rivera [13] made many neurons compete, modifying them by means of *fuzzy evolution*, i.e., using a fuzzy rules table that specifies the operator that must be applied to a neuron in order to improve its behavior.

Most recent methods still face single aspects when configuring a net. Thus, Ros et al.'s method [14] automatically initialize RBFNN, so that finds a good set of initial neurons, but relies in some other method to improve the net and to set its widths.

3 The EvRBF algorithm

EvRBF is an iterative procedure in which creation and evaluation of new generations of individuals leads to finding RBFNN with good generalization ability. It is a steady state algorithm, that includes elitism, and individuals size can vary while population size remains equal.

3.1 Genetic Operators

EvRBF provides operators to specifically deal with RBFNN, including: recombination or crossover, center and radii modification, and hidden layer size modification. Recombination allows to share information between individuals. Modification of centers, radii and number of neurons belong to the so-called *mutation operators*, and introduce a variability in the genetic pool. Following subsections describe these operators grouped on these three categories.

3.1.1 Recombination: X_FIX and X_MULTI

These operators interchange information between individuals, trying to find the *building blocks* of the solution. Thus, X_FIX reemplaces a sequence of hidden neurons of RBFNN R_1 by a sequence of the same size taken from RBFNN R_2. On the oher hand, operator X_MULTI reemplaces with probability p_{x_multi} every hidden neuron of RBFNN R_1 by a randomly chosen neuron coming from net R_2.

3.1.2 Centers and radii modification: C_RANDOM and R_RANDOM

These operators use randomness to increase diversity generating new individuals so that local minima can be avoided.

The C_RANDOM operator modifies RBF centers. The exact number of neurons affected by this operator is determined by its internal application probability: p_{c_random} (see subsection 3.3). C_RANDOM allows to explore the input space, since it swaps the current value, c_i, for a random value following an uniform probability function. Every component of the new central point, c_i', is taken from the range $[min_i, max_i]$ that are the minimum and maximum values of the $i - esime$ input space dimension. Both min_i and max_i can be calculated from available patterns.

On the other hand, R_RANDOM modifies the radii using the input space dimension width, and applying an uniform probability function. As in C_RANDOM, the real number of neurons affected by R_RANDOM depends on its internal application probability p_{r_random} (see 3.3 to know its current value).

3.1.3 Hidden layer size modification: ADDER and DELETER

EvRBF tries to determine the correct size of the hidden layer using the operator ADDER (that creates new neurons), and operator DELETER (that removes neurons).

ADDER adds a single neuron with random center and radii, using an uniform probability function in the range $[min_i, max_i]$. Selection pressure will reward or penalize nets containing the new neuron according to its effect in the fitness function.

On the other hand, DELETER removes neurons from the RBFNN to which is applied in a pure random way. The number of neurons to be removed depends on the value of p_{del_many}, that determines a neuron's probability of being removed (see subsection 3.3).

3.2 Rest of components

EvRBF incorporates usual mechanisms to select individuals for reproduction, to select operators to be applied, to extract information of the process; thus, the skeleton of EvRBF, showed in 1, is commonly used in evolutionary algorithms.

Training, validation and test sets are firstly used to estimate the input space ranges, as well as the input and output space dimensions. After this, training set is used to compute the synaptic weights (see section 1). Validation set is used to compute a fitness for every individual, once it has been trained. Training and validation sets do not share samples. Test set is used at the end of training to qualify every individual in the last generation.

According to [6], input values have been rescaled to $[0, 1]$ range. This way, every input dimension has the same weight when computing the distance between the RBF center and the point being evaluated.

1. Load training, validation and test sets.
2. Create, train, evaluate and set fitness of first generation.
3. While stop condition is not reached, do the following:
   (a) Select and copy individuals from current population.
   (b) Apply operators, train, evaluate and assign a fitness to that copies.
   (c) Replace worst individuals by the these copies
4. Train individuals in last generation with training and validation data sets.
5. Use test data set to obtain the generalization ability of every individual.

Figure 1: General skeleton of EvRBF.

Fitness function measures the generalization capability of the individual as the percentage of samples it correctly classifies. When comparing two individuals, the one with the higher generalization ability is considered better.

The population size used by EvRBF is constant. Every individual is a complete RBFNN, and the data structure used to represent it includes methods to be created, copied, and destroyed, to access and modify its components, and to compute an output value from an input pattern. First population is created using a given process (see subsection 3.3), subsequent generations are created by removing individuals from the population and adding copies of existing individuals to which operators have been applied.

Stop conditions included by EvRBF check the number of generations, error threshold, and individuals convergence. They can be used in combination or independently. In this work, the algorithm has been stopped when a pre-specified number of generations has been reached. This allows a better comparison with methods found in literature.

Finally, *selection, reproduction and substitution* are applied to populations. The selection operator chooses individuals to be reproduced. EvRBF uses fixed-length tournament method as selection operator, giving a small opportunity to bad individual to reproduce. Reproduction consist on copying the selected individual and, then, applying operators. Substitution sorts individuals according to their fitness, removes the worst of them, and adds the new individuals generated by reproduction.

3.3 Execution parameters

EvRBF needs to be given a set of parameters to work properly. Currently, a series of systematic experiments are being carried out to establish the best value that should be assigned to every parameter, although this study falls out of the scope of this paper. Thus, for the set of experiments described in section 4, EvRBF has been executed usign estandard values commonly used in literature. Parameters considered are shown in tables 1 and 2.

Parameter	*Value*
Population size	100
Generations	$\{10, 20, 50, 75, 100\}$
Max. neurons on first population	10% of patterns in training set.
Center initialization	Random patterns from training set
Radii initialization	Weighted distance between neurons
Replaced population	30%
Tournament length	2

Table 1: Execution parameters for EvRBF. See also table 2.

Table 1 shows that nets in the first generation can have a random number of hidden neuron, going from 1 to 10% of training set size. After this generation, no limit to the number of neurons is imposed.

Furthermore, centers are initialized using randomly selected patterns from the training; radii are setting as 1.75 times the minimum distance found between the centers of the net; finally, in every generation, 30 individuals are replaced by 30 new individuals created by reproduction.

Operator	*Application rate*	*Internal application prob.*
X_FIX	0.4	-
X_MULTI	0.4	0.5
C_RANDOM	0.05	0.5
R_RANDOM	0.05	0.5
ADDER	0.05	-
DELETER	0.05	0.5

Table 2: Application rates and internal application probabilities for operators.

Table 2 shows *application rate* and *internal application probability* for every operator. The values are set according to commonly used values in literature,

this is, 0.4 for crossover operator, and 0.05 for mutation operators. Internal application probabilities are used to decide which neurons from the RBFNN will be affected by the action of the operator (they correspond to p_{x_multi}, $p_{x\_average}$, p_{c_random}, p_{r_random}, and $p_{deleter}$, cited along subsection 3.1). For this work, these values have been set to 0.5 so that half the neurons of a net are affected in average.

4 Experiments and results

Evaluation of EvRBF on classification tasks has been developed using three well known real databases: the Iris plants, heart disease and cancer [1]. The increasing number of inputs in every problem shows how dimensionality affects to EvRBF.

Every problem has been run independently for different numbers of generations: 10, 25, 50, 75 and 100. For any of these values, the algorithm has been run 10 times, starting with different initial populations, and using different training and validation sets. Results related to EvRBF, shown in following subsections, are averaged values along the 10 runs per number of generations, taking into account only the best individual created in every execution.

4.1 Iris plants

Iris database patterns are composed of 4 input numeric values, and 1 output corresponding to the kind of plant (3 different plants being considered). Every class is represented by about 50 patterns. There are no null values and patterns are correctly classified. The data base has 156 patterns divided in 50% for test, $37,5\%$ for training and $12,5\%$ for validation.

EvRBF is compared to Whitehead and Choate's method [16], and Burdsall and Giraud's methods [2], since both of them use evolutionary algorithm to create RBFNN that solve this problem. Furthermore, [2] provides results yielded by an hand-optimized RBFNN (referred by *RBF* in table 3), by a nearest-attracting prototype (NAP) classifier, and by a multilayer perceptron (MLP).

Table 3 shows that EvRBF classifies better (0.0% error compared to Whitehead and Choate's 3%) using a small number of neurons (between 6 and 8). Second best method is Whitehead and Choate's one (an evolutionary algorithm too); Burdsall and Giraud's method does not improve results obtained by the hand-optimized RBFNN; and, finally, MLP and NAP provide the worst results. No information about minima and maxima obtained by EvRBF is going to be shown along this work; nevertheless, nets that correctly classified the whole test set have been found for this particular problem.

Algorithm		*Error %*	*RBFNN Size*
Whitehead-Choate [16]		3	15
Burdsall-Giraud [2]		4	6-9
RBF[2]		4	-
MLP[2]		4	-
NAP [2]		4	-
EvRBF	*10 gen.*	**0.0** $\pm$ 1	7 $\pm$ 1
EvRBF	*25 gen.*	0.6 $\pm$ 1	7 $\pm$ 2
EvRBF	*50 gen.*	1.4 $\pm$ 1	8 $\pm$ 1
EvRBF	*75 gen.*	0.8 $\pm$ 1	**6** $\pm$ 1
EvRBF	*100 gen.*	0.6 $\pm$ 1	7 $\pm$ 1

Table 3: Comparison of methods on Iris database classification

Algorithm	*Error %*	*Net size*
Prechelt[4]	1.15	-
Grönroos[4]	2.0	-
Quick-Propagation[4]	4.0$\pm$2	38
SA-Prop[4]	1.1$\pm$0.5	-
G-Prop[4]	**1.0**$\pm$0.5	**14**
EvRBF 10 gen.	1.7$\pm$0.3	30 $\pm$ 9
EvRBF 25 gen.	1.7$\pm$0.0	27 $\pm$ 10
EvRBF 50 gen.	1.7$\pm$0.5	36 $\pm$ 6
EvRBF 75 gen.	2.0$\pm$0.3	30 $\pm$ 13
EvRBF 100 gen.	3.7$\pm$1.2	26 $\pm$ 11

Table 4: Comparison of methods on Cancer-1a database classification

4.2 Cancer-1a database

Cancer-1a database, initially proposed by Prechelt [12], is related to breast cancer diagnosis. The problem consists of determining whether a given cancer is malign or benign, according to the analysis of cells via electronic microscopy. Every pattern is composed of 9 inputs and 1 output (0 -benign-, 1 -malign). 65.5% of patterns belong to benign tumors. The training set contains 524 patterns, while the test set contains 124. This is the problem initially considered by Grönroos [10].

Results from EvRBF are compared to those yielded by G-Prop [4] and others (Prechelt, Grönroos, Quick-Propagation and SA-Prop) cited in the same paper. G-Prop is an evolutionary algorithm similar to EvRBF, but designed to evolve multilayer perceptrons.

In table 4 can be seen the different percentages of error yielded by the methods of Prechelt and Grönroos, as well as the multilayer perceptrons trained with Quick-Propagation, or evolved by SA-Prop and G-Prop Except those related to EvRBF, results have been borrowed from [4].

In general, EvRBF's behaves is as good as the rest of methods being considered. Worst result yielded by EvRBF (3.7%) is slightly better than Quick-

Propagation (4.0%), although the rest of its results are similar or better than Grönroos (2.0%) and Prechelt (1.15%). In fact, this problem shows EvRBF's ability to tune the neurons (as can be seen in results obtained for 10, 25 and 50 generations) instead of adding many new neurons. This ability is mainly due to the genetic operators used in EvRBF, since fitness function depends on classification ability without applying any kind of penalization factor to the size of nets.

EvRBF yields the best results for 10, 25 and 50 generations. This result similar to all the previously cited, but worse than Castillos's SA-Prop (1.1%) and G-Prop (1.0%). The latter is also the best according to net sizes: it can solve the problem with only 14 neurons, instead of between 20 and 36 used by EvRBF or 38, used by Quick-Propagation.

4.3 Heart Disease Database

The last problem concerns diagnosis of heart disease, and has been taken from [1]. This database is composed of 270 samples, each one with 13 input variables (7 of them taking numerical values, 3 binary values and 3 nominal values) and 1 output class (two possible values indicating presence or absence of heart disease). Training set includes 165 samples, validation set has 55 samples and test set has only 50 samples. There are almost the same number of samples belonging to any of the possible classes.

This database allows comparison of EvRBF with Leonardis and Bischof's method [8], intended to automatically generate RBFNN. This pruning method can modify centers, radii and weight by means of gradient descent, and uses MDL to remove neurons from an initial over-specified net.

Table 5 shows results obtained by EvRBF for this problem. Columns show the percentage of patterns badly classified, and nets' sizes. As can be seen, the classification error and network size increase with the number of generations, taking their maximum values for 50 generations. Thus, for this problem, the algorithm produces overfitted nets that iteratively improve their results over the training and validation sets, while reducing their generalization capabilities.

For this problem, EvRBF took about 5 times more than for Iris database to reach the final generation. This is due to the higher dimension of the input space, what leads to a bigger number of neurons affected by operators, as well as a larger cost on computing the synaptic weights.

Table 5 compares results yielded by EvRBF, Leonardis-Bischof's method [8], and Gaussian Masking [15] (GM; it uses clustering and lineal programming to build neural nets), whose results have been taken from [8]. Results show that EvRBF is slightly worse by the rest of algorithms, although it uses smaller nets. Furthermore, it shows that as the number of generation grows, the nets become overfitted (see rows belonging to 50, 75 and 100 generations on table 5), making the generalization ability of the nets almost half of Leonardis and Bischof's nets and GM's ones.

Algorithm	*Error %*	*RBFNN size*
Leonardis-Bishop[8]	14	16
Gaussian Masking[8]	12	24
EvRBF 10 gen.	18.6 ± 3.5	11 ± 2
EvRBF 25 gen.	18.8 ± 3.7	12 ± 2
EvRBF 50 gen.	24.0 ± 5.3	10 ± 4
EvRBF 75 gen.	21.2 ± 2.4	14 ± 2
EvRBF 100 gen.	22.1 ± 1.9	15 ± 2

Table 5: Comparison of methods on Heart Disease database classification

4.4 General remarks

Setting the individuals fitness using only their classification skills leads some times to RBFNN with high generalization ability but an excessive number of neurons. Thus, keeping EvRBF as general as possible can lead to overspecified nets in some problems, but makes it suitable to be directly used in any kind of classification problem. Furthermore, no decisions have to be taken a priori concerning to the structure of the net, or to the zone of search space where solutions are supposed to be. EvRBF provides good solutions that can also be used as entries to local search algorithms in order to tune them.

5 Conclusion

This papers shows EvRBF, an algorithm that evolves RBFNN, applied to the task of solving classification problems.

EvRBF automatically determines both topology and configuration of RBFNN. Thus, EvRBF finds the size of the net (number of hidden neurons) and the parameters that configure each neuron: center and radii of its activation function. EvRBF does not start from pre-established topologies, does not set number of neurons a priori, does not fix an upper limit to the number of neurons, and uses only information about generalization capability and size of nets to evaluate and evolve them.

EvRBF includes operators specifically designed to deal with RBFNN. Operators can do cross-over, addition, removing and modification of hidden neurons, as well as their components. The parameters needed

by the algorithm to apply the operators have been empirically set to fixed values, and are used in every problem. Thus, no explicit pre-processing is needed to apply the algorithm to new problems.

In this paper, EvRBF algorithm has been applied to different real examples. Results have been compared to others in literature showing that can be used to efficiently configure RBFNN, since it obtains nets with high generalization ability and size smaller or similar to other methods.

The weakest point of the algorithm may be considered the number of parameters that configure the final nets. This number is very often greater than desired (referred to the number of values composing the training and test pattern sets). This is mainly due to the number of radii that EvRBF has to evolve: $2n$ per neuron, where n is the input dimension. Reducing this number to n radii per neuron (i.e., using symmetric RBF), or a single radius per neuron, or even a single radius per net would drastically reduce the number of parameters while probably maintaining a similar or slightly worst capability of generalization. Although this could be easily introduced into the algorithm, the aim of this work was to fully configure the nets, and deal with the widest range of RBF; thus, genetic operators are intended to look for the optimal set of values for all the parameters.

Future lines of research will focus on a deeper analysis of the parameters to run the algorithm (by means of ANOVA method), the use of cross-validation to set the fitness of every new individual, as well as facing this problem as a multiobjective optimizaztion task. Finally, we are currently obtaining the data necessary to apply this method to labor risk prevention.

Acknowledgements: This work has been partially supported by projects TIN2005- 08386-C05-03, and RNM-475.

References:

[1] C.L. Blake and C.J. Merz., *UCI Repository of machine learning databases*, 1998.

[2] B. Burdsall and C. Giraud-Carrier. *GA-RBF: A Self- Optimising RBF Network*. In ICANNGA'97, p. 348-351. Springer-Verlag, 1997.

[3] B. Carse and T.C. Fogarty. *Fast evolutionary learning of minimal radial basis function neural networks using a genetic algorithm*. In Evolutionary Computing, pp. 1-22. Springer-Verlag, 1996.

[4] P.A. Castillo, J. González, J.J. Merelo, V. Rivas, G. Romero and A. Prieto, *G-Prop-II: Global Optimization of Multilayer Perceptrons using GAs*, In CEC'99, pp. 2022-2027, USA, 1999.

[5] P.A. Castillo et al. *G-Prop: Global optimization of multilayer perceptrons using GAs*. Neurocomputing, 35:149-163, November 2000.

[6] W.S. Sarle. *Neural Network FAQ. Newsgroup: comp.ai.neural-nets*. Part 2, 1998. ftp://ftp.sas.com/pub/neural/FAQ.html.

[7] C. Harpham et al. *A review of genetic algorithms applied to training radial basis function networks*, Neural Computing & Applications 13 (3) (2004) 193-201

[8] A. Leonardis, H. Bischof, *And efficient MDL-based construction of RBF networks*, Neural Networks 11 (1998) 963-973.

[9] D. Broomhead, D. Lowe, *Multivariable Functional Interpolation and Adaptative Networks*, Complex Systems 11 (1988) 321-355.

[10] M. Grönroos,*Evolutionary Design of Neural Networks, Master of Science Thesis in Computer Science*. Dep. of Mathematical Sciences. University of Turku.

[11] J. Merelo, A. Prieto, *G-LVQ, a combination of genetic algorithms and LVQ*, in Artificial Neural Nets and Genetic Algorithms, pp. 92-95, Springer-Verlag, 1995.

[12] L. Prechelt, PROBEN1 – A set of benchmarks and benchmarking rules for neural network training algorithms, Tech. Rep. 21/94, Fakultät für Informatik, Universität Karlsruhe, D-76128 Karlsruhe, Germany (Sep. 1994).

[13] A. Rivera, J. Ortega, M. del Jesus, J. González, *Aproximación de funciones con evolución difusa mediante cooperación y competición de RBFs*, in AEB'02, pp. 507-514, 2002.

[14] F. Ros, M. Pintore, A. Deman, J. R. Chrétien, *Automatical initialization of RBF neural networks*, Chemometrics and intelligent laboratory systems. doi: 10.1016/j.chemolab.2006.01.008.

[15] A. Roy, S. Govil, R. Miranda, *An Algorithm to Generate Radial Basis Function (RBF)-Like Nets for Classification Problems*, Neural Networks 8 (2) (1995) 179- 201.

[16] B. A. Whitehead, T. Choate, *Cooperative-Competitive Genetic Evolution of Radial Basis Function Centers and Widths for Time Series Prediction*, IEEE Trans. on Neural Networks 7 (4) (1996) 869-880.