Neural Networks 24 (2011) 779–784

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Neural networks letter

Evolutionary q-Gaussian radial basis function neural networks for multiclassification

Francisco Fernández-Navarro [a,*], César Hervás-Martínez [a], P.A. Gutiérrez [a], M. Carbonero-Ruz [b]

[a] *Department of Computer Science and Numerical Analysis, University of Córdoba, Campus de Rabanales, Albert Einstein Building, 3rd floor, 14074 - Córdoba, Spain*

[b] *Department of Management and Quantitative Methods, ETEA, Escritor Castilla Aguayo 4, 14004 - Córdoba, Spain*

ARTICLE INFO

Article history:
Received 30 April 2010
Received in revised form 25 February 2011
Accepted 8 March 2011

Keywords:
q-Gaussian radial basis function neural networks
Evolutionary algorithm
Multiclassification
Hybrid algorithm

ABSTRACT

This paper proposes a radial basis function neural network (RBFNN), called the q-Gaussian RBFNN, that reproduces different radial basis functions (RBFs) by means of a real parameter q. The architecture, weights and node topology are learnt through a hybrid algorithm (HA). In order to test the overall performance, an experimental study with sixteen data sets taken from the UCI repository is presented. The q-Gaussian RBFNN was compared to RBFNNs with Gaussian, Cauchy and inverse multiquadratic RBFs in the hidden layer and to other probabilistic classifiers, including different RBFNN design methods, support vector machines (SVMs), a sparse classifier (sparse multinomial logistic regression, SMLR) and a non-sparse classifier (regularized multinomial logistic regression, RMLR). The results show that the q-Gaussian model can be considered very competitive with the other classification methods.

© 2011 Elsevier Ltd. All rights reserved.

1. Introduction

Different kinds of neural networks are being used for classification purposes, including multilayer perceptron neural networks (MLPNNs) in which the transfer functions are sigmoidal unit basis functions (Haykin, 2008), radial basis function neural networks (RBFNNs) with kernel functions in which the transfer functions are usually Gaussian (Bishop, 1996) and product unit neural networks (PUNNs) (Martínez-Estudillo, Hervás-Martínez, Gutiérrez, & Martínez-Estudillo, 2008), with multiplicative units.

In contrast to MLPs and PUNNs, RBFNNs use a localized representation of information. There are several common kinds of functions used as the transfer functions, for example, standard Gaussian (SRBF), multiquadratic (MRBF), inverse multiquadratic (IMRBF), and Cauchy (CRBF) ones. In this paper, we investigate the performance of the q-Gaussian RBFNN in multiclassification problems. This type of RBF can reproduce different RBFs, by updating a real parameter q, and allowing different shapes of RBFs in the same neural network.

Traditionally, an iterative training algorithm (e.g. a gradient-based algorithm) or clustering methods in combination with linear optimization techniques (e.g. k-means techniques and singular value decomposition) are applied to find the parameters of an RBFNN. Most learning algorithms proposed for constructing RBFNNs conduct a clustering analysis on the training data set and allocate one hidden unit for each cluster (Hwang & Bang, 1997; Musavi, Ahmed, Chan, Farms, & Hummels, 1992). In recent years, there has been growing interest in optimizing the radial unit parameters of RBFNNs using evolutionary algorithms (Patrinos, Alexandridis, Ninos, & Sarimveis, 2010; Pérez-Godoy, Fernández, Rivera, & Del Jesus, 2010).

Evolutionary computation algorithms have also been used for selecting variables for RBFNNs (Billings, Wei, & Balikhin, 2007) and for enhancing RBFNN training (Rivas, Merelo, Castillo, Arenas, & Castellano, 2004). Evolutionary algorithms (EAs) generally require a great number of iterations, and they converge slowly, especially in the neighbourhood of the global optimum. It thus makes sense to incorporate a faster local search (LS) algorithm into the EA in order to overcome this lack of efficiency while retaining the advantages of both optimization methods. In the machine learning community, these kinds of algorithms are known as hybrid algorithms (HAs).

The non-linearity of the RBFs with respect to the parameters implies that the corresponding Hessian matrix is generally indefinite, and the likelihood function could have a local optimum. In our opinion, these reasons justify the use of an alternative heuristic procedure to obtain the optimized parameters of the q-Gaussian RBFNN model.

The rest of this paper is organized as follows. A brief analysis of the q-Gaussian distribution is given in Section 2. The q-Gaussian RBF NN is presented in Section 3. Section 4 describes base classifier applied to multiclassification problems. A methodology for optimizing the RBF parameters based on HAs is presented in Section 5. Section 6 explains the experiments that were carried out. Finally, Section 7 summarizes the conclusions of our work.

[*] Corresponding author. Tel.: +34 957 21 83 49; fax: +34 957 21 83 60.
E-mail address: i22fenaf@uco.es (F. Fernández-Navarro).

0893-6080/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2011.03.014

2. The *q*-Gaussian distribution

Distributions known as normal or Gaussian, exponential, Laplace, etc. can be obtained through the principle of maximum entropy (Jaynes, 1957), under certain constraints, together with the normalization condition of probability (Kang & Kwak, 2009). The entropy function used in this approach is called the Boltzmann–Gibbs–Shannon (BGS) entropy, defined as follows: $S_{\mathrm{BGS}} = \int_{\Omega} p(x) \ln p(x) \mathrm{d}x$, where x is a random variable belonging to a certain set $\Omega \in \mathbb{R}$, $p(x)$ is the probability density function (pdf), and $p(x)\mathrm{d}x$ is the probability that the system is in states x and $x+\mathrm{d}x$.

However, some alternatives have emerged for replacing the BGS traditional entropy function. Among them, one of the most attractive has been proposed by Tsallis (1988): $S_q = \frac{1-\int_{\Omega} p(x)^q \mathrm{d}x}{q-1}$, where $q \in \mathbb{R}$. When $q = 1$, it reproduces the BGS entropic form. The mathematical basis for Tsallis statistics includes q-generalized expressions for the logarithm and exponential functions, which are the q-logarithm and the q-exponential functions. The q-exponential function, which reduces to $\exp(x)$ in the limit of $q \to 1$, is defined as follows:

$$e_q^x \equiv (1 + (1-q)x)^{\frac{1}{1-q}} = \frac{1}{(1-(q-1))^{\frac{1}{q-1}}}. \quad (1)$$

The q-distributions can arise when the exponential function of the original distribution is replaced by a q-exponential function. This basic procedure applied to a standard Gaussian distribution leads to a q-Gaussian distribution. This viewpoint suggests that other q-distributions should be considered.

In this way, the q-Gaussian distribution is obtained by replacing the exponential function by a q-exponential function and maximizing the entropy S_q under the following constraints (Tsallis, Mendes, & Plastino, 1998): (a) $\int_{\Omega} p(x)\mathrm{d}x = 1$, (b) $\int_{\Omega} xP_{esc}(x)\mathrm{d}x = \mu_q$ and (c) $\int_{\Omega}(x-\mu)^2 P_{esc}\mathrm{d}x = \sigma_q^2 > 0$, where $P_{esc}(x)$ is the escort probability, defined as

$$P_{esc}(x) = \frac{p^q(x)}{\int_{\Omega} p^q(x)\mathrm{d}x}. \quad (2)$$

Therefore, the q-Gaussian distribution is specified by the following pdf ($-\infty < q < 3$):

$$\begin{aligned} p(x, \mu_q, \sigma_q) &= A_q\sqrt{B_q}[1 + (q-1)B_q(x-\mu_q)^2]^{1/1-q} \\ &= A_q\sqrt{B_q} e_q^{-B_q(x-\mu_q)^2} \end{aligned} \quad (3)$$

where the parameters A_q, and B_q are defined as follows. The normalization factor A_q is given by

$$A_q = \begin{cases} \dfrac{\Gamma\left[\frac{5-3q}{2(1-q)}\right]}{\Gamma\left[\frac{2-q}{1-q}\right]}\sqrt{\dfrac{1-q}{\pi}} & q < 1; \\ \dfrac{1}{\sqrt{\pi}} & q = 1; \\ \dfrac{\Gamma\left[\frac{1}{q-1}\right]}{\Gamma\left[\frac{3-q}{2(q-1)}\right]}\sqrt{\dfrac{q-1}{\pi}} & 1 < q < 3. \end{cases} \quad (4)$$

Finally, the width of the distribution is characterized by

$$B_q = [(3-q)\sigma_q^2]^{-1}, \quad q \in (-\infty, 3). \quad (5)$$

In the limit of $q \to 1$, Eq. (3) recovers the usual Gaussian distribution form, so $q \neq 1$ indicates a departure from Gaussian statistics. For $3 \leq q$, the form given in Eq. (3) is not normalizable. When $q = 2$, the q-Gaussian distribution reproduces the Cauchy distribution. The usual variance (second-order moment) is finite for $q < 5/3$, and, for the standard q-Gaussian distribution ($N_q(0, 1)$), is given by $\sigma^2 = (3-q)/(5-3q)$. The usual variance of the q-Gaussian distribution diverges for $5/3 \leq q < 3$; however the q-variance remains finite for the full range $-\infty < q < 3$, equal to unity for the standard q-Gaussian distribution.

An example of an application of the q-Gaussian distribution can be seen in Erdemir and Tanatar (2003), where the q-Gaussian distribution was tested as a wavefunction for studying the properties of high density Bose–Einstein condensates. On the other hand, the q-Gaussian distribution has been employed in the study of a wide range of themes including Bose-condensed gases (Nicolin & Carretero-González, 2008) and DNA molecules (Moreira, Albuquerque, da Silva, & Galvao, 2008).

On the basis of the idea of the q-Gaussian distribution, we define the q-Gaussian RBF, by transforming the exponential expression of the standard RBF to a q-exponential expression.

3. *q*-Gaussian radial basis function neural networks

We focus on RBFNNs (Billings et al., 2007) which have been successfully employed in different pattern recognition problems in the last few years. Let the number of nodes in the input layer, in the hidden layer and in the output layer be K, M and J respectively. For any sample $\mathbf{x} = [x_1, x_2, \ldots, x_K]$, the output of the RBFNN is $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_J(\mathbf{x})]$. The model of an RBFNN can be described with the following equation:

$$f_j(\mathbf{x}) = \beta_{0j} + \sum_{i=1}^{M} \beta_{ij} \cdot \phi_i(d_i(\mathbf{x})), \quad j = 1, 2, \ldots, J \quad (6)$$

where $\phi_i(d_i(\mathbf{x}))$ is a non-linear mapping from the input layer to the hidden layer, $\boldsymbol{\beta}_j = [\beta_{1j}, \beta_{2j}, \ldots, \beta_{Mj}]$, for $j = 1, 2, \ldots, J$, is the weight of connection between the hidden layer and the output layer, and β_{0j} is the bias value for the class j. The function $d_i(\mathbf{x})$ can be defined as $d_i(\mathbf{x}) = \frac{\|\mathbf{x}-\mathbf{c}_i\|^2}{r_i^2}$, where r_i is the scalar parameter that defines the width for the ith radial unit, $\|.\|$ represents the Euclidean norm and $\mathbf{c}_i = [c_1, c_2, \ldots, c_K]$ are the centres of the RBFs. The standard RBF (SRBF) is the Gaussian function, which is given by $\phi_i(d_i(\mathbf{x})) = e^{-d_i(\mathbf{x})}$.

SRBFs present a very selective response, with high activation for patterns close to the centroid and very small activation for distant patterns. The RBFs $\phi_i(d_i(\mathbf{x}))$ can take different forms, including the Cauchy RBF (CRBF) form defined by $\phi_i(d_i(\mathbf{x})) = \frac{1}{1+d_i(\mathbf{x})}$ and the inverse multiquadratic RBF (IMRBF) form, given by $\phi_i(d_i(\mathbf{x})) = \frac{1}{(1+d_i(\mathbf{x}))^{\frac{1}{2}}}$.

The CRBF and the IMRBF have longer tails than the SRBF, i.e., their activations for patterns distant from the centroid of the RBF are bigger than the activation of the SRBF for those patterns. In addition, the SRBF, CRBF and IMRBF functions do not fall asymptotically to zero.

Other functions used in place of the SRBF as RBFs, could be piecewise linear functions (PLRBFs) (Wang, Lu, & Chen, 2010), cubic approximations (Stein & Feuer, 1998) and the thin plate spline functions (TPSRBFs) which were employed in Dehghan and Shokri (2008) to solve the two-dimensional damped/undamped sine–Gordon equation. The radial cubic B-spline was introduced in Saranli and Baykal (1998), where they concluded that these functions achieve very similar performance to the SRBF, because radial cubic B-splines and SRBFs have similar convergence properties.

In this paper, we investigate the use of the q-Gaussian RBF for multiclassification problems because this family of functions considers, as already discussed in the previous section, different kinds of local functions, in which the tails of the different functions

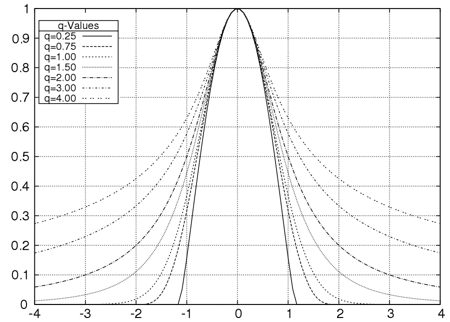

Fig. 1. Radial unit activation in one-dimensional space with $c = 0$ and $\theta = 1$ for q-Gaussian RBFs with different values of q.

play crucial roles and one can reduce the q-Gaussian function to the standard Gaussian function. The q-Gaussian RBF for the RBF j can be defined as $\phi_i(d_i(\mathbf{x})) = e_{q_j}^{-d_j(\mathbf{x})}$, where q_j is a real valued parameter and the q-exponential function of $-d_j(\mathbf{x})$ is given by

$$\phi_i(d_i(\mathbf{x})) = \begin{cases} (1-(1-q)d_i(\mathbf{x}))^{\frac{1}{1-q}} & \text{if } (1-(1-q)d_i(\mathbf{x})) \geq 0; \\ 0 & \text{Otherwise.} \end{cases} \tag{7}$$

The q-Gaussian RBF can reproduce different RBFs for different values of the real parameter q: when the q parameter is close to 2, the q-Gaussian is the CRBF; for $q = 3$ we have the activation of a radial unit with an IMRBF for $d_i(\mathbf{x})$ equal to the activation of a radial unit with a q-Gaussian RBF for $d_i(\mathbf{x})/2$; and, finally, when the value of q converges to 1, the q-Gaussian converges to the SRBF. Fig. 1 presents the radial unit activation for the q-Gaussian RBF for different values of q.

4. *q*-Gaussian RBFs for multiclassification

In a classification problem, measurements x_i, $i = 1, 2, \ldots, K$, of a single individual (or object) are taken, and the individuals are to be classified into one of the J classes on the basis of these measurements. A training sample $D = \{(\mathbf{x}_n, \mathbf{y}_n); n = 1, 2, \ldots, N\}$ is available, where $\mathbf{x}_n = (x_{1n}, \ldots, x_{kn})$ is the random vector of measurements taking values in $\Omega \subset \mathbb{R}^K$, and $\mathbf{y}_n$ is the class level of the nth individual, where the common technique of representing class levels using a "1-of-J" encoding vector is adopted, $\mathbf{y} = (y^{(1)}, y^{(2)}, \ldots, y^{(J)})$.

In order to tackle this classification problem, the outputs of the q-Gaussian RBFNN model have been interpreted from the point of view of probability through the use of the softmax activation function:

$$g_l(\mathbf{x}, \boldsymbol{\theta}_l) = \frac{\exp f_l(\mathbf{x}, \boldsymbol{\theta}_l)}{\sum_{j=1}^{J} \exp f_j(\mathbf{x}, \boldsymbol{\theta}_j)}, \quad l = 1, 2, \ldots, J \tag{8}$$

where $f_j(\mathbf{x}, \boldsymbol{\theta}_l)$ (Eq. (6)) is the output of the jth output neuron for pattern $\mathbf{x}$ and $g_l(\mathbf{x}, \boldsymbol{\theta}_l)$ is the probability that a pattern $\mathbf{x}$ has of belonging to class j.

The function used to evaluate a q-Gaussian RBFNN is the function of cross-entropy error and it is given by the following expression:

$$\begin{aligned} l(\boldsymbol{\theta}) &= -\frac{1}{N} \sum_{n=1}^{N} \sum_{l=1}^{J} y_n^{(l)} \log g_l(\mathbf{x}, \boldsymbol{\theta}_l) \\ &= \frac{1}{N} \sum_{n=1}^{N} \left[-\sum_{l=1}^{J} y_n^{(l)} f_l(\mathbf{x}_n, \boldsymbol{\theta}_l) + \log \sum_{l=1}^{J} \exp f_l(\mathbf{x}_n, \boldsymbol{\theta}_l) \right] \end{aligned} \tag{9}$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J)$. A scheme for these models is given in the website associated with this paper.[1]

```
Hybrid Algorithm:
Generate a random population of size $N$
repeat
    Calculate the fitness of every individual in the population
    Rank the individuals with respect to their fitness
    The best individual is copied into the new population
    The best 10% of population individuals are replicated
      and they substitute the worst 10% of individuals
    Apply parametric mutation to the best $(p_m)$% of individuals
    Apply structural mutation to the remaining $(100-p_m)$%
      of individuals
until the stopping criterion is fulfilled
Apply $iRprop+$ to the best solution obtained by the EA in
  the last generation.
```

Fig. 2. Hybrid algorithm (HA) framework.

5. Hybrid algorithms

The basic framework of the HA is the following: the search begins with an initial population of RBFNNs and, in each iteration, the population is updated using a population-update algorithm that evolves both its structure and its weights. The population is subject to operations of replication and mutation. The neural networks are represented using an object-oriented approach and the algorithm deals directly with the RBFNN phenotype. Fig. 2 describes the procedure used to select the parameters of the q-Gaussian RBFNN. The main characteristics of the algorithm are the following:

1. *Error and fitness functions*. We consider $l(\boldsymbol{\theta})$ (Eq. (9)) as the error function of an individual g of the population. The fitness measure needed for evaluating the individuals is a strictly decreasing transformation of the error function $l(\boldsymbol{\theta})$ given by $A(\boldsymbol{\theta}) = \frac{1}{1+l(\boldsymbol{\theta})}$, where $0 < A(\boldsymbol{\theta}) \leq 1$.
2. *Initialization of the population*. First, 5000 random RBFNNs are generated. The centres of the radial units are first defined using the k-means algorithm for different values of k, where $k \in [M_{\min}, M_{\max}]$, and $M_{\min}$ and $M_{\max}$ are the minimum and maximum number of hidden nodes allowed for any RBFNN model in the HA. The widths of the RBFNNs are initialized to the geometric mean of the distance to the two nearest neighbourhoods and the q parameter to values near to 1, because when $q \to 1$ the q-Gaussian RBF reduces to the SRBF. A random value in the $[-I, I]$ interval is assigned for the weights between the hidden layer and the output layer. The individuals obtained are evaluated using the fitness function and the initial population is finally obtained by selecting the 500 best RBFNNs.
3. *Structural mutation*. There are four different structural mutations: hidden node addition, hidden node deletion, connection addition and connection deletion. These four mutations are applied sequentially to each network, each one with a specific probability. If the structural mutator adds a new node in the RBFNN, the q parameter is assigned to a γ value, where $\gamma \in [0.75, 1.25]$.
4. *Parametric mutation*. Different weight mutations were applied:
   - *Centre, radius and q mutation*. These parameters were modified in the following way:
     - Centre creep. The value of each centre is modified by adding a Gaussian noise term, $c_{ji}(t+1) = c_{ji}(t) + \xi_1(t)$,

[1] http://www.uco.es/ayrna/QRBF.

where $\xi_1(t) \in N(c_{ji}, r_i)$ and $N(c_{ji}, r_i)$ represents a one-dimensional normally distributed random variable with mean c_{ji} and with a standard deviation equal to the radius of the ith RBF hidden node.

- Radius creep. The value of each radius is modified by adding another Gaussian noise, $r_i(t+1) = r_i(t) + \xi_2(t)$, where $\xi_2(t) \in N(r_i, d)$ and $N(r_i, d)$ represents a one-dimensional normally distributed random variable with mean r_i and with standard deviation the width of the range of each dimension (d).
- Mutation of the q parameter. The q parameter is updated by adding a uniform ε value, where $\varepsilon \in [-0.25, 0.25]$.

- *Output-to-hidden node connection mutations*. These connections are modified by adding another Gaussian noise term, $w(t+1) = w(t) + \xi(t)$, where $\xi(t) \in N(0, T(g))$ and $N(0, T(g))$ represents a one-dimensional normally distributed random variable with mean 0 and variance equal to the network temperature ($T(g) = 1 - A(g)$). Further details on these kinds of mutations can be found in Hervás-Martínez, Martínez-Estudillo, and Carbonero-Ruz (2008).

5. *iRprop+ local optimizer*. The local optimization algorithm used in our paper is the *iRprop+* (Igel & Hüsken, 2003) optimization method. In the proposed methodology, we run the EA and then apply the local optimization algorithm to the best solution obtained by the EA in the last generation. The adaptation of the *iRprop+* local improvement procedure can be seen in the website associated with this paper.

6. Experiments

The proposed methodology was applied to sixteen data sets[2] taken from the UCI repository (Asuncion & Newman, 2007). All nominal variables were transformed to binary variables. The data sets with their corresponding partitions have been included in the website associated with this paper.

6.1. Experimental design

The proposed method (q-Gaussian) is compared with the following:

- Other RBFs obtained with the same HA (detailed in 5):
  - The standard radial basis functions (SRBF) where the transfer function is Gaussian.
  - The Cauchy radial basis function (CRBF).
  - The inverse multiquadratic radial basis function (IMRBF).
- Some high performance probabilistic classifiers:
  - A Gaussian RBF network (RBFN) (Nabney, 2004), deriving the centres and width of hidden units using the k-means approach and combining the outputs obtained from the hidden layer using logistic regression.
  - The AdaBoost.M1 algorithm (Freund & Schapire, 1996), using an RBFN as the base learner and the maximum number of iterations set to 100 iterations (Ada100(RBFN)).
  - The C-SVM algorithm (Hastie, Tibshirani, & Friedman, 2001) with Gaussian RBF kernels (SVM).
  - The sparse multinomial logistic regression (SMLR) algorithm (Krishnapuram, Carin, Figueiredo, & Hartemink, 2005). This method has been selected as a good representative of recently developed sparse classifiers (RVM, PCVM, ...).
  - The regularized multinomial logistic regression (RMLR) algorithm (Yamashita, Sato, Yoshioka, Tong, & Kamitani, 2008): a multiclass version of the RLR (regularized logistic regression with Laplace approximation) algorithm.

For the selection of the SVM hyperparameters (the regularization parameter, C, and the width of the Gaussian functions, γ), a grid search algorithm was applied with a tenfold cross-validation, using the following ranges: $C \in \{2^{-5}, 2^{-3}, \dots, 2^{15}\}$ and $\gamma \in \{2^{-15}, 2^{-13}, \dots, 2^{3}\}$. RMLR and SMLR algorithms estimate the λ regularization value automatically.

All the parameters used in the HA except the maximum and minimum numbers of RBFs in the hidden layer ($[M_{\min}, M_{\max}]$) and the number of generations ($\#Gen$) have the same values in all problems analysed below. For the selection of these parameters, a grid search algorithm was applied with a tenfold cross-validation in an analogous way to the method used in the algorithms that it is compared with, using the following ranges: $[M_{\min}, M_{\max}] \in \{[2, 5], [4, 7], [9, 12]\}$ and $\#Gen \in \{20, 40, 100, 400\}$.

We did a simple linear rescaling of the input variables over the interval $[-2, 2]$, with X_i^* being the transformed variables. The connections between the hidden and output layer were initialized in the $[-5, 5]$ interval (i.e. $[-I, I] = [-5, 5]$). The size of the population was $N = 500$. For the structural mutation, the number of nodes that could be added or removed was within the $[1, 2]$ interval, and the number of connections to add or delete in the hidden and the output layer during structural mutations was within the $[1, 7]$ interval.

For the models obtained by the HA proposed in this paper (q-Gaussian, SRBF, IMRBF and CRBF), the experimental design was conducted using a tenfold cross-validation, with ten repetitions per part. For the other methods, the results were obtained by performing a tenfold cross-validation ten times, because they were all deterministic methods. The performance of each method was evaluated using the correct classification rate (C) in the generalization set.

The HA and the model proposed were implemented in Java. For the other RBFs (the CRBF, SRBF and IMRBF), the *iRprop+* algorithm was modified slightly, taking into account which RBF was being used in the hidden layer. We also used "libsvm" (Chang & Lin, 2001) to obtain the results from the SVM method, and WEKA to obtain the results from the RBFN and Ada100(RBFN) methods. The SMLR and RMLR methods belong to the SLR toolbox, available as a suite of MATLAB functions and scripts.[3]

6.2. Analysis of the results

In this section, we analyse the results obtained. Specifically, we check the performance (mean accuracy value from the 100 executions of each data set) of the GRBF model and eight other related methodologies. For the sake of simplicity, we only include the graphical and statistical results achieved; the complete results can be found at the website associated with this paper.

In the scatterplot of Fig. 3, each point compares GRBF to another methodology on a single data set. The x-axis position of the point is the accuracy of the GRBF, and the y-axis position is the accuracy of the compared algorithm. Therefore, points below the $y = x$ line correspond to data sets for which GRBF performs better in mean than the other algorithm. From the analysis of the results, it can be concluded that the q-Gaussian model produced the best mean ranking ($\bar{R} = 2.00$) and reported the highest mean accuracy ($\bar{C}_G = 84.99\%$).

To determine the statistical significance of the rank differences observed for each method in the different data sets, we carried out a non-parametric Friedman test (Friedman, 1940) with the ranking

[2] Data set titles: Hepatitis, Heart-disease, Breast-cancer, Heart, Liver, Vote, Card, German, Wine, Newthyroid, Horse, Balance, Lymphography, Anneal, Glass, Zoo.

[3] http://www.cns.atr.jp/~oyamashi/SLR_WEB/.

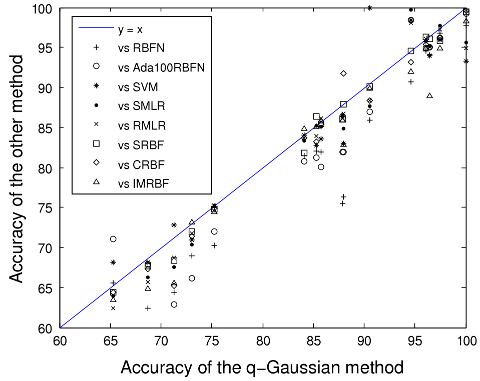

Fig. 3. Comparison of the proposed basis functions to other methods: accuracy results over 16 data sets.

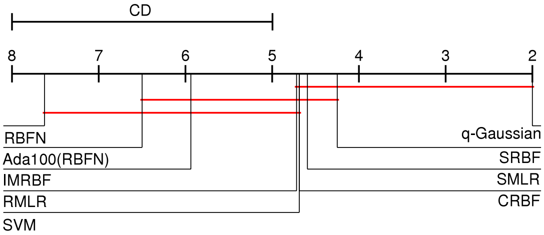

Fig. 4. The Nemenyi test using C_G as the variable test. CD is the critical difference.

of C_G of the best models as the test variable. The test showed that the effect of the method used for classification was statistically significant at a significance level of 5%.

On the basis of this rejection, the Nemenyi post hoc test was used to compare all classifiers with each other (Hochberg & Tamhane, 1987). However, it has been noted that the approach of comparing all classifiers with each other in a post hoc test is not as sensitive as the approach of comparing all classifiers to a given classifier (a control method). One approach to this latter type of comparison is the Holm test.

The results of the Holm and Nemenyi tests (Fig. 4) for $\alpha = 0.05$ can be seen at the website associated with this paper. Note that q-Gaussian model is established as the control algorithm because it has obtained the best mean ranking. Using a level of significance $\alpha = 0.05$, the q-Gaussian is significantly better than the rest of the methods, considering the accuracy measure, which justifies the proposal.

6.3. Analysis of the best q-Gaussian model for the Liver data set

In this section, we study in detail the best q-Gaussian RBFNN obtained for the biclass Liver data set. We considered the best model to be of one of the ten folds used in the experiments (specifically the ninth one). The model for the Liver data set was determined by three basis functions:

$$\phi_1(d_1) = (1 - (1 - 0.106) \cdot d_1(\mathbf{x}))^{\frac{1}{1-0.106}}$$

$$\phi_2(d_2) = (1 - (1 - 1.024) \cdot d_2(\mathbf{x}))^{\frac{1}{1-1.024}}$$

$$\phi_3(d_3) = (1 - (1 - 1.077) \cdot d_3(\mathbf{x}))^{\frac{1}{1-1.077}}$$

where the function $d_i(\mathbf{x})$ for $i = 1, 2, 3$ is defined as

$$d_1 = \left(\frac{\sqrt{(x_1^\star - 0.601)^2 + (x_3^\star + 1.253)^2 + (x_5^\star + 2.053)^2}}{0.523}\right)^2$$

$$d_2 = \left(\frac{\sqrt{(x_1^\star - 1.692)^2 + (x_3^\star - 0.076)^2 + (x_4^\star + 0.758)^2}}{1.287}\right)^2$$

$$d_3 = \left(\frac{\sqrt{(x_4^\star - 0.629)^2}}{1.498}\right)^2$$

$$x_i^\star \in [-2, 2] \quad \text{for } i = 1, \ldots, 6$$

and the output of the softmax transformation is

$$\widehat{g_1}(\mathbf{x}) = \frac{\exp(-0.723 + 3.501\phi_1 + 5.226\phi_2 - 2.646\phi_3)}{1 + \exp(-0.723 + 3.501\phi_1 + 5.226\phi_2 - 2.646\phi_3)}.$$

By using the properties of softmax, the decision rule can be expressed in a more simplified way using the discriminant function $C(\mathbf{x})$:

$$C(\mathbf{x}) = \begin{cases} 1 & \text{if } 3.501\phi_1 + 5.226\phi_2 - 2.646\phi_3 > 0.723 \\ 0 & \text{if } 3.501\phi_1 + 5.226\phi_2 - 2.646\phi_3 < 0.723. \end{cases}$$

As we can see, the best model for the Liver data set is composed of two standard RBFs ($\phi_2(d_2(\mathbf{x}))$ and $\phi_3(d_3(\mathbf{x}))$) and a non-standard RBF ($\phi_1(d_1(\mathbf{x}))$ where the q value is 0.106). The activation of the q-Gaussian RBF $\phi_1(d_1(\mathbf{x}))$ for patterns distant from the centroid was smaller than the activation of the remaining RBFs.

The performance of this q-Gaussian model on the training set was 71.38% and on the generalization set it was 85.29%. It is important to note that, if this model was of the SRBF type ($q \to 1$), the performance of the model on the training set was 64.30% and on the generalization set it was 76.47%.

One of the major advantages of the q-Gaussian model is the reduced number of features and RBFs included in the final expression, because the HA reduces its complexity by pruning mutations. This can result in a better interpretability of the model, which is especially important when dealing with real problems.

On the other hand, we observed that the q-Gaussian model transformed the six-dimensional input space into a three-dimensional space given by the basis functions. The model is aimed at capturing the interactions among the variables and reducing the dimensionality of the space. It is interesting to note that this reduction allowed us to depict the separation of the two classes into training (Fig. 5(a)) and generalization points (Fig. 5(b)) by means of linear functions in the transformed space. Finally, Fig. 5(c)

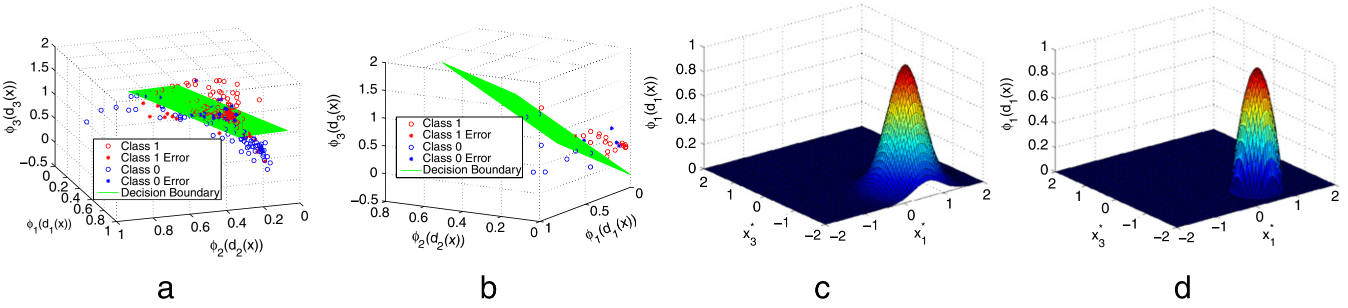

Fig. 5. Analysis of the performance of the q-Gaussian RBFNN on the Liver data set: graphics for (a) training and (b) generalization points and the decision boundary; and graphics for $\phi_1(d_1(\mathbf{x}))$ using the variables x_1^* and x_3^* when (c) $q \to 1$ (SRBF) and (d) q has been optimized by the HA ($q = 0.106$).

and (d) represent the $\phi_1(d_1(\mathbf{x}))$ using the standard RBF (the q-Gaussian with $q \to 1$) and the corresponding q-Gaussian with the q value optimized by the HA.

7. Conclusions

In this paper, we proposed a new approach for determining the optimized parameters for the q-Gaussian RBFNN. The use of q-Gaussian RBFs made it possible to modify the shape of the RBF by changing the real parameter q. The q-Gaussian RBFNN proposed used the softmax function and the cross-entropy error function in order to interpret the output of the q-Gaussian RBFNN from the point of view of probability. The coefficients that minimized the cross-entropy error function were estimated by means of an HA. The large experimental study performed allowed us to show that this proposal is a suitable method for addressing multiclassification problems.

Acknowledgement

This work was partially subsidized by the TIN 2008-06681-C06-03 project of the Spanish Inter-Ministerial Commission of Science and Technology (MICYT), FEDER funds and the P08-TIC-3745 project of the "Junta de Andalucía" (Spain). The research of Francisco Fernández-Navarro was funded by the "Junta de Andalucia" Predoctoral Programme, grant reference P08-TIC-3745.

References

Asuncion, A., & Newman, D. (2007). UCI machine learning repository. URL http://www.ics.uci.edu/~mlearn/MLRepository.html.

Billings, S. A., Wei, H., & Balikhin, M. A. (2007). Generalized multiscale radial basis function networks. *Neural Networks*, *20*(10), 1081–1094.

Bishop, C. M. (1996). *Neural networks for pattern recognition*. Oxford, UK: Oxford University Press.

Chang, C., & Lin, C. (2001). Libsvm: a library for support vector machines.

Dehghan, M., & Shokri, A. (2008). A numerical method for solution of the two-dimensional sine–Gordon equation using the RBF. *Mathematics and Computers in Simulation*, *79*(3), 700–715.

Erdemir, E., & Tanatar, B. (2003). q-Gaussian trial function in high density Bose–Einstein condensates. *Physica A: Statistical Mechanics and its Applications*, *322*, 449–455.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proceedings of the thirteenth international conference on machine learning* (pp. 148–156). Morgan Kaufmann.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of *m* rankings. *Annals of Mathematical Statistics*, *11*(1), 86–92.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). *The elements of statistical learning*. Springer.

Haykin, S. (2008). *Neural networks: a comprehensive foundation* (3rd ed.). Prentice Hall.

Hervás-Martínez, C., Martínez-Estudillo, F. J., & Carbonero-Ruz, M. (2008). Multilogistic regression by means of evolutionary product-unit neural networks. *Neural Networks*, *21*(7), 951–961.

Hochberg, Y., & Tamhane, A. (1987). *Multiple comparison procedures*. John Wiley & Sons.

Hwang, Y., & Bang, S. (1997). An efficient method lo construct radial basis function neural network classifier. *Neural Networks*, *10*(8), 1495–1503.

Igel, C., & Hüsken, M. (2003). Empirical evaluation of the improved rprop learning algorithms. *Neurocomputing*, *50*(6), 105–123.

Jaynes, E. T. (1957). Information theory and statistical mechanics. *Physical Review*, *106*(4), 620–630.

Kang, H. Y., & Kwak, B. M. (2009). Application of maximum entropy principle for reliability-based design optimization. *Structural and Multidisciplinary Optimization*, *38*(4), 331–346.

Krishnapuram, B., Carin, L., Figueiredo, M. A. T., & Hartemink, A. J. (2005). Sparse multinomial logistic regression: fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(6), 957–968.

Martínez-Estudillo, F. J., Hervás-Martínez, C., Gutiérrez, P. A., & Martínez-Estudillo, A. C. (2008). Evolutionary product-unit neural networks classifiers. *Neurocomputing*, *72*(1–2), 548–561.

Moreira, D. A., Albuquerque, E. L., da Silva, L. R., & Galvao, D. S. (2008). Low-temperature specific heat spectra considering nonextensive long-range correlated quasiperiodic DNA molecules. *Physica A: Statistical Mechanics and its Applications*, *387*(22), 5477–5482.

Musavi, M. T., Ahmed, W., Chan, K. H., Farms, K. B., & Hummels, D. M. (1992). On the training of radial basis function classifiers. *Neural Networks*, *5*, 595–603.

Nabney, I. T. (2004). Efficient training of rbf networks for classification. *International Journal of Neural Systems*, *14*(3), 201–208.

Nicolin, A. I., & Carretero-González, R. (2008). Nonlinear dynamics of Bose-condensed gases by means of a q-Gaussian variational approach. *Physica A: Statistical Mechanics and its Applications*, *387*(24), 6032–6044.

Patrinos, P., Alexandridis, A., Ninos, K., & Sarimveis, H. (2010). Variable selection in nonlinear modeling based on rbf networks and evolutionary computation. *International Journal of Neural Systems*, *20*(5), 365–379.

Pérez-Godoy, M. D., Fernández, A., Rivera, A. J., & Del Jesus, M. J. (2010). Analysis of an evolutionary rbfn design algorithm, co2rbfn, for imbalanced data sets. *Pattern Recognition Letters*, *31*(15), 2375–2388.

Rivas, V. M., Merelo, J. J., Castillo, P. A., Arenas, M. G., & Castellano, J. G. (2004). Evolving rbf neural networks for time-series forecasting with evrbf. *Information Sciences*, *165*(3–4), 207–220.

Saranli, A., & Baykal, B. (1998). Complexity reduction in radial basis function (rbf) networks by using radial b-spline functions. *Neurocomputing*, *18*(1–3), 183–194.

Stein, D., & Feuer, A. (1998). Cubic approximation neural network for multivariate functions. *Neural Networks*, *11*(2), 235–248.

Tsallis, C. (1988). Possible generalization of Boltzmann–Gibbs statistics. *Journal of Statistical Physics*, *52*(1–2), 479–487.

Tsallis, C., Mendes, R. S., & Plastino, A. R. (1998). The role of constraints within generalized nonextensive statistics. *Physica A: Statistical Mechanics and its Applications*, *261*(3–4), 534–554.

Wang, L., Lu, W., & Chen, T. (2010). Coexistence and local stability of multiple equilibria in neural networks with piecewise linear nondecreasing activation functions. *Neural Networks*, *23*(2), 189–200.

Yamashita, O., Sato, M., Yoshioka, T., Tong, F., & Kamitani, Y. (2008). Sparse estimation automatically selects voxels relevant for the decoding of fmri activity patterns. *NeuroImage*, *42*(4), 1414–1429.