Combining genetic optimisation with hybrid learning algorithm for radial basis function neural networks

Lin Guo, De-Shuang Huang and Wenbo Zhao

A two-step learning scheme for radial basis function neural networks (RBFNN) is proposed. A genetic algorithm initially optimises the parameters of the RBFNN and a hybrid learning algorithm adjusts these parameters further. The designed network is not only parsimonious but also has better generalisation performance.

Introduction: The determination of the hidden centres and the radial basis function widths is of particular importance to the optimisation of radial basis function neural networks (RBFNN). There are many approaches to determine the hidden centres. One is the clustering method, including input clustering [1] and input–output clustering [2]. These two methods usually result in a large number of selected centres. Another approach is the recursive orthogonal least square algorithm (ROLSA) [3]. Although this algorithm is of fast convergent speed, it cannot avoid local minima and easily produces suboptimal solutions. In addition, a genetic optimisation method is proposed in [4]. Although a parsimonious network can be achieved using this method, the better generalisation performance cannot usually be guaranteed. In this Letter we propose a novel method combining genetic optimisation with the hybrid learning algorithm (HLA) [5] to optimise the RBFNN.

RBFNN: An RBFNN can be considered as a mapping: $\Re^r \rightarrow \Re^M$. Let $\mathbf{P} \in \Re^r$ be the input vector and $\mathbf{C_i} \in \Re^r (1 \le i \le u)$ be the hidden centres. Usually, the Gaussian function is preferred among all radial basis functions. Hence, the output of each RBF neuron is:

$$R_i(\mathbf{P}) = \exp\left(-\frac{\|\mathbf{P} - \mathbf{C_i}\|_2}{\sigma_i^2}\right) \tag{1}$$

where $\|\cdot\|_2$ indicates the Euclidean norm in the input space and σ_i is the width of the ith RBF. Usually, the same pattern class neuron has the same width. The jth output, $y_j(\mathbf{P})$, of an RBFNN is:

$$y_j(\mathbf{P}) = \sum_{i=1}^{u} R_i(\mathbf{P}) \times w(j, i) \tag{2}$$

where $w(j, i)$ is the weight from the jth output to the ith hidden neuron.

GA and HLA: A general framework for the genetic algorithm (GA) has been described in [6]. In the framework, the key issues are how to encode a solution as a chromosome, how to define a fitness function to evaluate each individual, and how to apply genetic operators to the encoded chromosome.

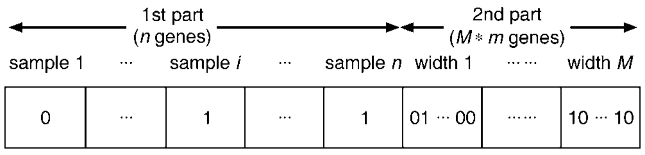

Fig. 1 *Encoding scheme of individual*

We adopt the similar genetic encoding method of chromosome described in [4]. Specifically, to globally design an optimised RBFNN, the RBF widths should also be selected by the GA. As shown in Fig. 1, each gene is encoded into a binary bit and each individual includes two encoding parts. The first part has n genes encoding the centres and the second part has $M \times m$ genes encoding the widths, where M denotes the pattern class number (i.e. the number of output neurons of the RBFNN) and m is the coding length of one width parameter. In the first part each gene only represents a fixed sample. We make use of binary symbols, 1's or 0's, to denote whether the training samples are selected as the centres of the hidden neurons or not. Our objective is to make the number of hidden centres selected as few as possible under the given accuracy. So the fitness function value monotonously decreases with the increase of the number of hidden centres. When the practical output error exceeds the given error, the fitness function value reaches the minimum (mostly zero) regardless of how large the number of hidden centres is. Thus, we design a fitness function as follows:

$$f(\varepsilon, n_c) = \frac{(b+1)n}{2^b n_c}, \quad b = sign(\varepsilon - e) \tag{3}$$

where ε represents the given error in advance; e is the practical root mean squared errors (RMSE) of the output layer; n is the total number of the training samples and n_c the number of the selected hidden centres. In addition, we adopt the same genetic operators as in [4].

The HLA, combining the gradient paradigm and the linear least square (LLS) paradigm, can be used to adjust the centres and the widths. Generally, effective initialisation of the centres and the widths is required. This algorithm includes two passes. In the forward pass, we supply input data and functional signals to calculate the hidden output, $\mathbf{R}$. Then, the weight $\mathbf{W}$ is modified by the LLS method as follows:

$$\mathbf{W}^* = \mathbf{D}({}^T_{\mathbf{R}}\mathbf{R})^{-1}{}^T_{\mathbf{R}} \tag{4}$$

where ${}^T_{\mathbf{R}}$ is the transpose of $\mathbf{R}$, and $\mathbf{D}$ the target matrix consisting of 1's and 0's. After identifying the weight, the functional signals continue going forward until the error measure is calculated. In the backward pass, the errors propagate from the output end towards the input end. Keeping the weight fixed, the centres and widths of the RBF neurons are modified as follows:

$$E^l = \frac{1}{2}\sum_{k=1}^{M}(d_k^1 - y_k^1)^2 \quad l = 1, 2, \ldots, n \tag{5}$$

$$\begin{aligned}\Delta C^l(i,j) &= -\xi \frac{\partial E^l}{\partial C^l(i,j)} \\ &= -\xi \frac{\partial E^l}{\partial y_k^l}\frac{\partial y_k^l}{\partial R_j^l}\frac{\partial R_j^l}{C^l(i,j)} \quad i = 1, 2, \ldots r, \ j = 1, \ldots, u \\ &= 2\xi \sum_{k=1}^{s}(d_k^l - y_k^l) \cdot w^l(k,j) \cdot R_j^l \cdot \frac{P(i,l) - C^l(i,j)}{(\sigma_j^l)^2}\end{aligned} \tag{6}$$

$$\begin{aligned}\Delta\sigma_j^l &= -\xi \frac{\partial E^l}{\partial \sigma_j^l} = -\xi \frac{\partial E^l}{\partial y_k^l}\frac{\partial y_k^l}{\partial R_j^l}\frac{\partial R_j^l}{\partial \sigma_j^l} \quad j = 1, \ldots, u \\ &= 2\xi \sum_{k=1}^{s}(d_k^l - y_k^l) \cdot w^l(k,j) \cdot R_j^l \cdot \frac{\|P_l - C^j\|^2}{(\sigma_j^l)^3}\end{aligned} \tag{7}$$

where E^l is the error function for the lth training pattern; d_k^l is the kth desired output for the lth training pattern; y_k^l is the kth actual output for the lth training pattern; $\Delta C^l(i, j)$ is the centre error rate of the ith input variable of the jth RBF unit for the lth training pattern; $\Delta\sigma_j^l$ is the width error rate of the jth RBF unit for the lth training pattern; $P(i, l)$ is the ith input variable for the lth training pattern and ξ the learning rate.

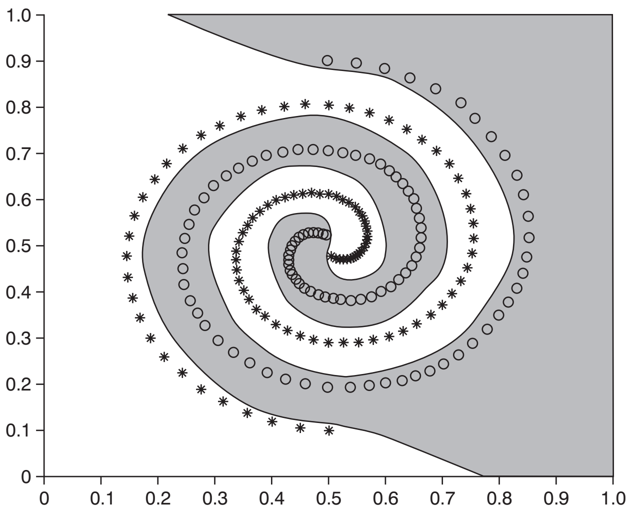

Fig. 2 *Telling-two-spirals-apart problem and its classification boundary with RBFNN trained by GA and HLA*

Simulation results: The proposed methods above are applied to solving the telling-two-spirals-apart problem [4]. The 200 training samples for this problem were produced according to [4]. The cross-over probability and the mutation probability with the GA are set as

$P_c = 0.8$ and $P_m = 0.002$, respectively. The given error criterion ε is set as 0.2. Consequently, after 500 generations, the selected hidden centres number is converged to 28, i.e. only 14% of the original training samples being needed, and the widths to $\sigma_1 = \sigma_2 = 0.15$. Compared to other methods, the selected centres numbers using the ROLSA and the k-means are 36 and 38, respectively. It can be seen that our method selected a fewer number of centres. We then take these parameters as the initial values of the centres and the widths for the HLA. After training the parameters of the RBFNN using the HLA, the corresponding RMSE decreases from 0.1954 to 0.0915. The classification boundary is plotted in Fig. 2. In addition, Table 1 shows the testing results of Gaussian white noisy training samples with zero means but different variances using various methods. From Table 1 and Fig. 2, it can be seen that the RBFNN learned by the HLA after initial optimisation by the GA has better generalisation capability.

Table 1: Recognition rate comparisons of various methods

Rooted variances of the noises	$\sigma = 0.01$	$\sigma = 0.025$	$\sigma = 0.05$	$\sigma = 0.075$	$\sigma = 0.1$
Recognition rates (GA)	97.5%	84.5%	71.0%	54.0%	53.5%
Recognition rates (k-means + HLA)	98.0%	85.5%	72.0%	54.5%	53.0%
Recognition rates (GA + HLA)	99.5%	89.5%	75.5%	55.0%	53.5%

Conclusion: A two-step learning scheme for the RBFNN is proposed. The GA was adopted to initially select the optimal hidden centres from the training samples and the widths of the RBF neurons. The HLA was then used to adjust those parameters. As a result, the experiments showed that a more efficient and parsimonious structure of RBFNN with better generalisation capability can be designed.

© IEE 2003 *2 July 2003*
Electronics Letters Online No: 20031021
DOI: 10.1049/el:20031021

Lin Guo, De-Shuang Huang and Wenbo Zhao (*Institute of Intelligent Machines, Chinese Academy of Sciences, P.O. Box 1130, Hefei Anhui 230031, People's Republic of China*)

E-mail: lguo@iim.ac.cn

References

1 MOODY, J.E., and DARKEN, C.J.: 'Fast learning in networks of locally tuned processing units', *Neural Comput.*, 1989, **1**, pp. 281–294
2 UYKAN, Z., and GUZELIS, C.: 'Input-output clustering for determining the centers of radial basis function networks'. Proc. ECCTD -97, Budapest, Hungary, 1997, **2**, pp. 435–439
3 GOMM, J.B., and YU, D.L.: 'Selecting radial basis function network centers with recursive orthogonal least squares Training', *IEEE Tran. Neural Netw.*, 2000, **11**, (3), pp. 306–314
4 ZHAO, W., and HUANG, D.S.: 'The structure optimization of radial basis probabilistic neural networks based on genetic algorithms'. Proc. JJCNN'02, Honolulu, HI, USA, May 2002, pp. 1086–1091
5 ER, M.J., *et al.*: 'Face recognition with radial basis function (RBF) neural networks', *IEEE Trans. Neural Netw.*, 2002, **13**, (3), pp. 697–710
6 HONG, S.G., *et al.*: 'Nonlinear time series modelling and prediction using Gaussian RBF network with evolutionary structure optimisation', *Electron. Lett.*, 2001, **37**, (10), pp. 639–640