

A Global Training Algorithm for Radial Basis Functions Neural Networks

Artur Ferreira (1) Mário Figueiredo (2)

(1) Centro de Cálculo e Departamento de Engenharia Electrónica das Telecomunicações e de Computadores
 Instituto Superior de Engenharia de Lisboa, 1950-062, Lisboa, Portugal

arturj@cc.isel.ipl.pt

(2) Instituto de Telecomunicações, Departamento de Engenharia Electrotécnica e de Computadores
 Instituto Superior Técnico, 1049-001 Lisboa, Portugal

mtf@lx.it.pt

Radial basis functions neural networks (RBFNN) are widely used for regression and classification problems. The topology of this neural network consists of an input layer, a hidden layer containing the radial basis functions and their centers, and an output layer performing the weighted sum of the values from the hidden layer.

The classical training algorithms for RBFNN are divided into two separate stages. The first stage consists of an unsupervised algorithm to adjust the centers of the hidden layer. After the first stage is complete, the second stage is carried out by a (supervised) least squares algorithm to adjust the weights of the output layer. These algorithms simplify the training problem by dividing it into two separate simpler problems. This simplification has the shortcoming that there is no interaction and adjustment between these stages.

In this work, we analyse the RBFNN as a mixture of Gaussians, considering that the radial basis functions, in the hidden layer, are Gaussians. We represent the Gaussian functions of the hidden layer by their mean vector and covariance matrix. This way, the RBFNN is treated as a global single model, to be trained globally. We derive an expectation-maximisation (EM) algorithm to learn the means and covariances of the Gaussian radial basis functions, from the training set. In each iteration of the EM algorithm, we simultaneously adjust the weights of the output layer by logistic regression. We exploit ways to combine EM and logistic regression algorithms. This way, the training of the RBFNN is performed with a global algorithm integrating EM and logistic regression.

Our approach is tested and compared against the two-stage training algorithms, on synthetic and real data for binary classification problems. In our tests, the proposed algorithm achieved better performance with faster training, when compared to two-stage algorithms.

Introduction

Radial basis functions neural networks (RBFNN) are widely used for classification and regression problems. Since their introduction, in the beginning of the decade of 1980, several training algorithms have been proposed and developed. In this work, we present a training algorithm which speeds up the training procedure and gives better performance than the conventional techniques.

This paper is organised as follows. First, we present the RBFNN topology and then we describe the existing two-stage training algorithms. The global training algorithm is then proposed and presented in detail. Comparative results between the two types of algorithm are presented and discussed. Based on these results, some conclusions are presented.

Radial Basis Functions Neural Networks

The RBFNN topology consists of an input layer, a hidden layer containing the radial basis functions and their centers, and an output layer which performs the weighted sum of the values from the hidden layer. Figure 1 shows the typical topology of an RBFNN, considering a bias term for each neuron in the output layer. Each

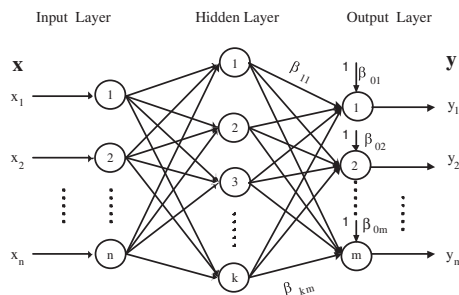


Figure 1: Typical topology of the RBFNN.

output y_i of the network is expressed as

$$y_i = g(\mathbf{x}) = \sum_{i=1}^k \beta_i G(\|\mathbf{x} - \mathbf{t}_i\|) \quad (1)$$

where $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ is the input, the β_i are the weights, $\|\cdot\|$ denotes some norm and \mathbf{t}_i are the $n \times 1$ centers (vectors). Associated with each neuron of the hidden layer there is a center \mathbf{t}_i and a Gauss function $G(a) = \exp\left(\frac{-a^2}{2}\right)$. In words, each RBFNN output results from a linear combination of the outputs of the hidden layer. Each output of the hidden layer is obtained by applying a Gauss function to the distance between the input \mathbf{x} and one of the centers \mathbf{t}_i .

Overview of existing training algorithms

Conventional training algorithms for RBFNN are usually divided into two separate stages [6]. The first stage learns the hidden layer while the second stage sets the output layer weights. In this section, we briefly overview several two-stage learning algorithms.

- Fixed centers selected at random [1, 6], in which centers are chosen randomly from the training set. Learns only the linear weights of the output layer, using the pseudo-inverse matrix (least-squares).
- Self-organised selection of centers. Centers are obtained by the KMeans [3, 6] algorithm and the output weights are estimated by LMS (Least Mean Squares) [6, 9]. The enhanced KMeans [2] algorithm, which attains the optimal or near optimal solution when compared to KMeans, can also be used.
- Supervised selection of centers. In this method, a supervised algorithm is used to learn both layers. A gradient descent algorithm is used with adaption formulas for the linear weights (output layer) and the position and spreads of the centers [6].
- Strict interpolation with regularisation [6, 15]. This algorithm, built upon regularisation and regression theories, has four phases: Nadaraya-Watson regression estimate, diagonal input norm matrix (for each center), regularised strict interpolation and selection of centers width and linear weights with LOO CV (Leave One Out Cross Validation) and GCV (General Cross Validation).
- The use of regression trees and decision trees are explored in [12] and [7], respectively. Regression trees are used to set the number, positions and sizes of all RBF centers. A decision tree, created by the C4.5 algorithm, maps to the centers.
- In [11], an EM algorithm is used on regularised RBFNN. It uses maximum marginal likelihood to the training data to establish critical parameters of the RBFNN for regression. The EM algorithm obtains the noise variance and prior weight variance, but not the basis functions widths.

Other training methods and algorithms have been proposed. In [10] regression trees and an EM algorithm are used to determine the weights, treating them as missing data, as the training data is considered as observed data. Another approach is taken in [14], in which the expectation-maximisation (EM) algorithm is applied to learn the entire network, interpreting learning as maximum likelihood problem. The idea to incorporate full or diagonal covariance matrices into RBFNN appeared in [8], named EBF (elliptical basis function) networks. The EM algorithm is used to learn the basis functions parameters and least squares determines the weights, on the second stage of the training. In [13] the basis functions are seen as probability densities and the weights as prior probabilities. The RBFNN training is seen as a maximum likelihood problem and an EM algorithm is used to learn the model parameters. Other approaches, interpretations and learning strategies can be found in [4].

The proposed global training algorithm

Our approach uses the EM algorithm to learn the hidden layer parameters while simultaneously updates the output layer coefficients. We analyse the RBF model as a mixture of Gaussians, considering that the radial basis functions, in the hidden layer, are Gaussians. We represent the Gaussian functions of the hidden layer by their mean vector and covariance matrix. This way, the RBF is treated as a global single model, to be trained globally. We derive an EM algorithm for learning the means and covariances of the Gaussian radial basis functions, from the training set. In each iteration of the EM algorithm, we simultaneously adjust the weights of the output layer by logistic regression [5]. We explore the combination of EM and logistic regression. This way, the training of the RBFNN is performed with a global algorithm consisting of EM and logistic regression.

Expectation-Maximisation (EM) algorithm

The EM algorithm is applied to estimate the mean vectors and covariance matrices of the Gauss functions in the hidden layer. We interpret the RBF as a mixture of Gaussians (MOG)

$$y_i = g(\mathbf{x}) = \sum_{i=1}^k \alpha_i \mathcal{N}(g_i | \mu_i, C_i), \quad (2)$$

where g_i is a data point, μ_i is the mean vector and C_i is the covariance matrix. For the Gaussian case we have the following equations for the E-Step and M-Step, respectively [3, 6].

E-Step - Computes the proportion (weight) of each Gaussian, over the entire set of Gaussians, with the vector

$$z_i = \frac{\hat{\alpha}_s^{(t)} \mathcal{N}(g_i | \hat{\mu}_s^{(t)}, \hat{C}_s^{(t)})}{\sum_{r=1}^k \hat{\alpha}_r^{(t)} \mathcal{N}(g_i | \hat{\mu}_r^{(t)}, \hat{C}_r^{(t)})} \quad (3)$$

M-Step - Updates the estimates of the α_i coefficients, mean vectors μ_i and covariance matrices C_i according to

$$\hat{\alpha}_s^{(t+1)} = \frac{1}{N} \sum_{i=1}^N z_i, \quad \hat{\mu}_s^{(t+1)} = \frac{\sum_{i=1}^N g_i z_i}{\sum_{i=1}^N z_i}, \quad \hat{C}_s^{(t)} = \frac{\sum_{i=1}^N (g_i - \hat{\mu}_s^{(t+1)})(g_i - \hat{\mu}_s^{(t+1)})^T z_i}{\sum_{i=1}^N z_i}. \quad (4)$$

In equations (3) and (4) we have $s \in \{1, 2, \dots, k\}$ and N is the number of data points. All these parameters are set to some initial values: the α_s coefficients are set to $\frac{1}{N}$, the vectors $\hat{\mu}_s$ have small values and are different from each other, and the C_s are set to the identity matrix.

Logistic regression

In order to learn the logistic regression model [5], that is, to learn the logistic regression coefficients β of the output layer (according to figure 1), we use the Newton-Raphson step

$$\beta \leftarrow \beta + (X^T W X)^{-1} X^T (y - p), \quad (5)$$

repeatedly, starting with $\beta = [0, \dots, 0]$, the $m \times 1$ vector with the logistic regression coefficients, y is a $N \times 1$ vector holding the class label, X is the $N \times m$ design matrix containing the training data passed through the model functions (Gaussians, in this case). The i th column of X contains the i th model function applied to the training data. Vector p has dimensions $N \times 1$ and contains the values of $p(x_i; \beta)(1 - p(x_i; \beta))$, where $p(x_i; \beta) = \text{logit}(X\beta)$, with $\text{logit}(a) = (1 + \exp(-a))^{-1}$. Finally, W is a $N \times N$ diagonal matrix, whose diagonal is the vector p . We have also considered the case of $W = 0.25 \times I_N$, (where I_N is the identity matrix), which is a majorant of the previous matrix. He have seen that produces faster convergence when compared to the previous matrix. Note that the matrix inversion in (5) is applied on a square matrix of dimensions m , the number of neurons in the hidden layer.

The algorithms

The idea behind the proposed training algorithms is the combination of EM with logistic regression and learning the hidden and the output layer parameters altogether. In this section, we detail the proposed global training algorithms, which are supposed to be carried out until a stopping criterion is met, like a maximum number of iterations or a given error rate, for example. The input to these algorithms are G , the $(n + 1) \times N$ matrix containing the training data and its class labels and k , the number of hidden neurons. The output are:

- μ_i and C_i , the $1 \times n$ mean vector and $n \times n$ covariance matrix of each Gaussian, $i \in \{1, \dots, k\}$.
- β - the $(m + 1) \times 1$ output layer coefficients.

EM-Log Algorithm

. Step 0 - Initialisation

$$\mu_i \leftarrow \text{random values in the range } [-1, 1], \quad C_i = I_n, \quad \beta = [0 \dots 0], \quad i \in \{1, \dots, k\}.$$

Step 1 - Train

Repeat steps 1a) to 1e) until stopping criterion is met:

1a) Compute the proportion of each center: $z_i = \frac{\hat{\alpha}_s \mathcal{N}(g_i | \hat{\mu}_s, \hat{C}_s)}{\sum_{r=1}^k \hat{\alpha}_r \mathcal{N}(g_i | \hat{\mu}_r, \hat{C}_r)}$

1b) Compute the weights of each center: $\hat{\alpha}_s = \frac{1}{N} \sum_{i=1}^N z_i$

1c) Update the mean estimate: $\hat{\mu}_s = \frac{\sum_{i=1}^N g_i z_i}{\sum_{i=1}^N z_i}$

1d) Update the covariance estimate: $\hat{C}_s = \frac{\sum_{i=1}^N (g_i - \hat{\mu}_s)(g_i - \hat{\mu}_s)^T z_i}{\sum_{i=1}^N z_i}$

1e) Update the logistic regression coefficients: $\beta \leftarrow \beta + (X^T W X)^{-1} X^T (y - p)$

EM-Log-Link Algorithm

The EM-Log-Link algorithm is like the EM-Log algorithm, adding a new step after 1e), named 1f) defined as

$$\alpha_i = \frac{|\beta_i|}{\sum_{i=1}^k |\beta_i|},$$

meaning that the EM α_i coefficients for the next iteration are updated according to the logistic regression coefficients.

Results

In this section, we compare EM-Log and EM-Log-Link with following two-stage algorithms:

- KMeans+LS - KMeans followed by Least Squares [5].
- KMeans+GR - KMeans followed by Global Ridge Regression [5].
- EM+LS - The output layer coefficients are updated by Least Squares, **only once**, after EM terminates.
- EM+Log - The output layer coefficients are updated by Logistic Regression, **only once**, after EM terminates.

We consider the case of binary classification, with a network topology of $n=2$, $k=4$ and $m=1$, as shown in figure 2. This network is trained with the two-stage and the global stage algorithms. In order to access the

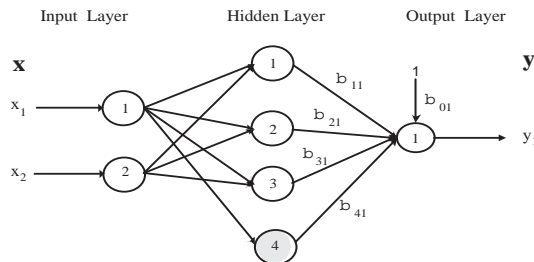


Figure 2: Topology of the network used for binary classification.

generalisation ability of the networks, we have also produced a test set, according to the training set. The first test set contains 400 points in two dimensions, from two Gaussian classes (200 points per class), while the training set has 200 points (100 per class). The data points, with classes marked with '+' and 'o' along with the classification results, on the test set, are shown in figure 3. The KMeans and EM algorithm have the same starting point: the initial centers for KMeans are equal to initial estimates of the mean vectors by EM. We evaluate the networks on the test set. Using $k=4$, and after 10 iterations both algorithms achieve 0 errors on the test set. Note the different classification areas, given by the level curves: for KMeans and the classical RBFNN we have a radial classification area, while for EM we have ellipsoidal areas, according to the covariance of the data points.

Now we consider another synthetic data set, with 100 data points on the training set and 200 data points on the test set. Figure 4 shows the classification results, on the test set, after 15 iterations, with $k=6$. We conclude that the EM-Log-Link algorithm achieves zero errors while KMeans+LS attains 15 classification errors. Other tests, showed that in the case of training data with uniform distribution, there is no appreciable differences between the algorithms. In the Gaussian case, the proposed algorithms are slightly superior. Finally, in situations

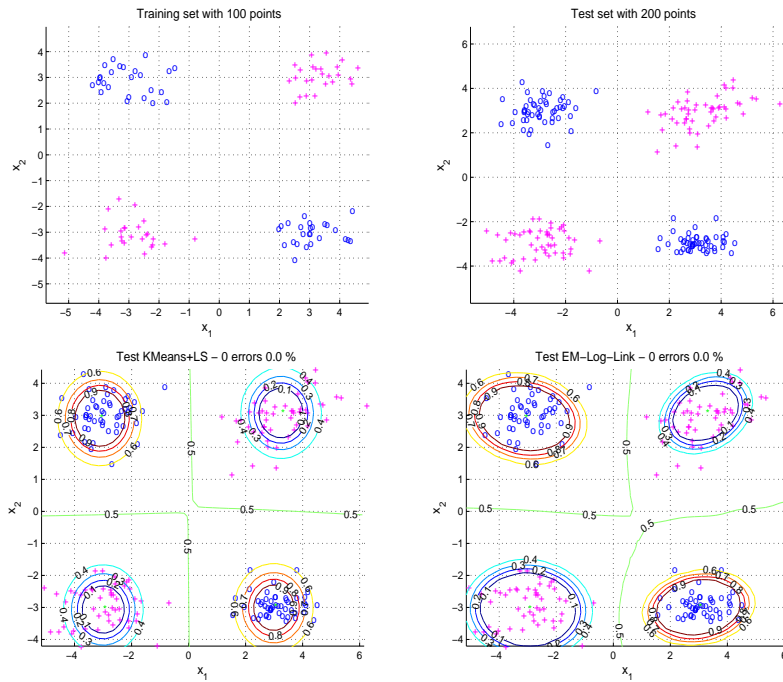


Figure 3: The training and test data sets and a comparison between the classification obtained by KMeans+LS and EM-Log-Link.

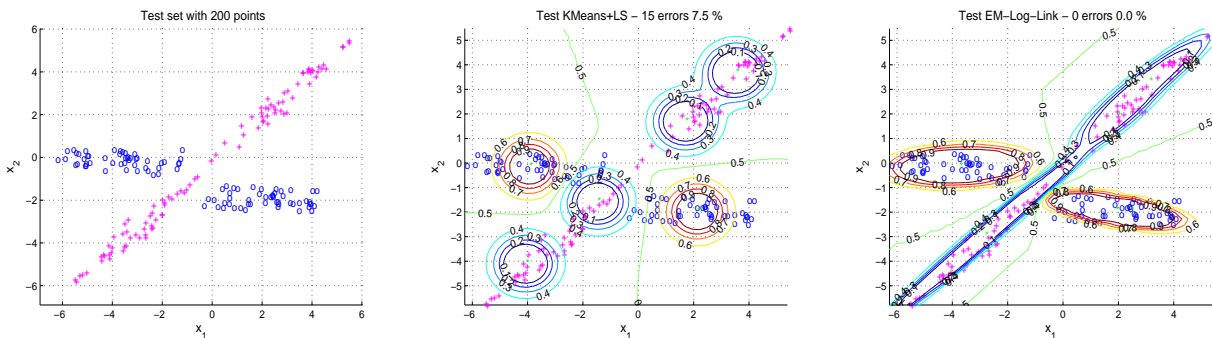


Figure 4: The test data set and the classification results obtained by KMeans+LS and EM-Log-Link.

as those reported in figure 4, the proposed algorithms converge faster, specially the EM-Log-Link algorithm.

Using the Ripley dataset, which has a Bayes error of 8%, we compare the two stage algorithms with the global algorithms, evaluating the number of classification errors on the test set. Using a topology with $k=6$ we get the results depicted in figure 5, after 2 training iterations. In this test, the EM+LS algorithm attains

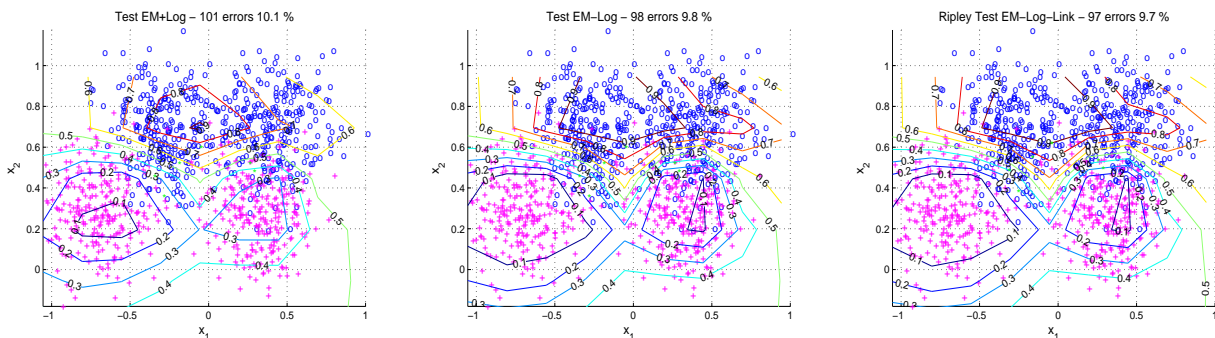


Figure 5: Comparison of EM+LOG, EM-Log and EM-Log-Link on the Ripley test set.

the same error rate as the EM+LOG algorithm. The EM-Log-Link algorithm performs slightly better than the EM-Log algorithm.

Conclusions

We have presented a global training algorithm for RBFNN, integrating EM and logistic regression; we have explored two methods to combine these two components. A comparison between the proposed algorithm and the conventional two-stage training algorithms was carried out for binary classification on synthetic and real data.

We conclude that the proposed global algorithm achieves faster training and better performance than the two-stage conventional approach. In the case that classes do not follow an uniform distribution, the proposed algorithm attains its best results. For the Gaussian case, our algorithm attains its best results when classes have different covariances with different orientations.

One critical issue is the initialisation of the mean vectors and covariance matrices. In this implementation, the mean vectors were initialised with a Gaussian distribution with zero mean. For the case of covariance matrices, we found that they should be initialised with small values; we chose the identity matrix. Future work includes the generalisation of these algorithms for regression.

References

- [1] D. BroomHead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.
- [2] C. Chinrungrueng and C. Séquin. Optimal adaptive KMeans algorithm with dynamic adjustment of learning rate. *IEEE Transactions on Neural Networks*, 6:157–169, 1994.
- [3] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2001.
- [4] M. Figueiredo. On Gaussian Radial Basis Functions Approximations: Interpretation, Extensions, and Learning Strategies. In *International Conference on Pattern Recognition*, volume 2, pages 618–621, Barcelona, September 2000.
- [5] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2nd edition, 2001.
- [6] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, N.J.: Prentice Hall, 2nd edition, 1999.
- [7] M. Kubat. Decision Trees can Initialize Radial-Basis Function Networks. *IEEE Transactions on Neural Networks*, 9(5):813–821, 1998.
- [8] M.-W. Mak and S.-Y. Kung. Estimation of elliptical basis function parameters by the EM algorithm with application to speaker verification. *IEEE Trans. on Neural Networks*, 11(4):961–969, July 2000.
- [9] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294, 1989.
- [10] M. Orr. Recent advances in radial basis function network. Technical report, Institute for Adaptive and Neural Computation, Division of Informatics, Edinburg, Scotland.
- [11] M. Orr. An EM-algorithm for regularised RBFN. In *International Conference on Neural Networks and Brain*, Beijing, China, 1998.
- [12] M. Orr. Combing regression trees with RBFN. *International Journal of Neural Systems*, 10:453–465, 2000.
- [13] M. Titsias and A. Likas. Shared kernel models for class conditional density estimation. *IEEE Trans. on Neural Networks*, 12(5):987–997, September 2001.
- [14] L. Xu. RBF nets, mixture experts, and Bayesian ying-yang learning. *Neurocomputing*, 19:223–257, 1998.
- [15] P. Yee. *Regularised radial basis function networks: theory and applications to probability estimation, classification, and time series prediction*. PhD thesis, McMaster University, Hamilton, Ontario, 1998.