ESANN'2006 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2006, d-side publi., ISBN 2-930307-06-4.

Hybrid Generative/Discriminative Training of Radial Basis Function Networks

Artur J. Ferreira[1] and Mário A. T. Figueiredo[2]

1- Instituto de Telecomunicações, Instituto Superior de Engenharia de Lisboa
1950-062 Lisboa, **Portugal**

2- Instituto de Telecomunicações, Instituto Superior Técnico
1049-001 Lisboa, **Portugal**

Abstract. We propose a new training algorithm for radial basis function networks (RBFN), which incorporates both generative (mixture-based) and discriminative (logistic) criteria. Our algorithm incorporates steps from the classical expectation-maximization algorithm for mixtures of Gaussians with a logistic regression step to update (in a discriminative way) the output weights. We also describe an incremental version of the algorithm, which is robust regarding initial conditions. Comparison of our approach with existing training algorithms, on (both synthetic and real) binary classification problems, shows that it achieves better performance.

1 Introduction

1.1 Basics of RBFN

RBFN are widely used for classification and regression problems. Since their introduction, several architectures and training algorithms have been proposed.

An RBFN consists of an input layer, a hidden layer containing the radial basis functions, and an output layer which performs the weighted sum of the values from the hidden layer. Each output y_i of a (linear) RBFN is given by

$$y_i = g(\mathbf{x}) = \beta_{i0} + \sum_{j=1}^{k} \beta_{ij}\, G(\|\mathbf{x} - \mathbf{t}_j\|), \tag{1}$$

where $\mathbf{x} = [x_1, \ldots, x_n]^T$ is the network input, β_{ij}, for $j = 1, ..., k$, is the weight between the j-th hidden neuron and the i-th output neuron, β_{i0} is the bias of the i-th output, $\|\cdot\|$ denotes some norm, $\mathbf{t}_j = [t_{j1}, ..., t_{jn}]^T$ is the center of the j-th RBF, and $G(a) = \exp(-a^2/2)$ is (usually) a Gauss function. Standard RBFN use the Euclidean norm, $\|\mathbf{x} - \mathbf{t}_j\| = \|\mathbf{x} - \mathbf{t}_j\|_2$, while elliptical basis function networks (EBFN) use Mahalanobis distances [1], based on "covariance-type" matrices $\|\mathbf{x} - \mathbf{t}_j\| = \|\mathbf{x} - \mathbf{t}_j\|_{\mathbf{C}_j} = ((\mathbf{x} - \mathbf{t}_j)^T \mathbf{C}_j^{-1} (\mathbf{x} - \mathbf{t}_j))^{\frac{1}{2}}$.

1.2 RBFN Training Algorithms

Early algorithms were two-stage procedures [2]: the first stage learns the hidden layer (the centers, and maybe the covariance matrices), while the second stage

sets the output weights. One of the most popular algorithms is the so-called *self-organized selection of centers* [2], which is based on K-means clustering [2, 3] on the first stage, followed by least squares (LS) adjustment of the output weights [2, 4]. The use of regression and decision trees is considered in [5] and [6], still in two-phase approaches. In [7], an expectation-maximization (EM) algorithm is used to learn some of the parameters of a regularized RBFN; however, this EM algorithm does not obtain the RBF widths and is only used for regression. Other approaches combining EM with regression trees are considered in [8], still only for regression problems. A two-stage approach for EBFN training is proposed in [1]: an EM algorithm estimates the centers and covariance matrices, followed by LS estimation of the output weights.

A global EM algorithm (which learns all the network parameters, including arbitrary "covariance" matrices) was proposed in [9], but only for regression problems. In [10], the basis functions are seen as probability densities and the weights as prior probabilities; global training is carried out by an EM algorithm which estimates the class-conditional densities. For the regression case, other global approaches, interpretations, and learning strategies can be found in [11]. General purpose optimization tools for non-convex problems (such as genetic algorithms and simulated annealing) have also been used for RBFN training.

1.3 Proposed Approach

In this paper, we propose a global EM-type algorithm to train an RBFN for classification; although extension to multi-class is trivial, we focus on the binary case. For classification purposes, we use a logistic RBFN obtained by applying a logistic function to the output of a linear RBFN with one output (see (1)). The output of the logistic RBFN is seen as the probability of class 1, given the input $\mathbf{x}$, that is,

$$P(y=1|\mathbf{x}) = \text{logistic}(g(\mathbf{x})) = \left[1 + \exp\left(-\beta_0 - \sum_{j=1}^{k} \beta_j \, G(\|\mathbf{x}-\mathbf{t}_j\|_{\mathbf{C}_j})\right)\right]^{-1}.$$

We propose to use the EM algorithm to learn the hidden layer parameters ($\mathbf{t}_j$ and $\mathbf{C}_j$), while simultaneously updating the output layer weights (β_j) by using logistic regression (LR).

This paper is organized as follows. Section 2 presents the proposed global training approach. Some results on synthetic and real binary classification problems are shown in Section 3. Finally, Section 4 presents some conclusions.

2 The Proposed Global Training Algorithms

The training set consists of N labelled data points, $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$, where $\mathbf{x}_i \in I\!R^n$ and $y_i \in \{0, 1\}$. The goal is to obtain all the parameters of the network: the centers $\mathbf{t}_j$ and covariances $\mathbf{C}_j$, for $j = 1, ..., k$, and the output weights $\boldsymbol{\beta} = [\beta_0, .., \beta_k]^T$.

By using normalized Gaussian basis functions, we can interpret the linear part of the RBFN as a mixture of Gaussians $g(\mathbf{x}) = \beta_0 + \sum_{i=1}^{k} \beta_i \mathcal{N}(\mathbf{x}|\mathbf{t}_i, \mathbf{C}_i)$. This clearly suggests using an EM algorithm for learning the means and covariances of the Gaussian basis functions. In each iteration of this EM algorithm, we simultaneously adjust the weights β_i of the output layer by logistic regression [12]. Notice that this algorithm can be seen as using a generative criterion for estimating the means and covariances, and a discriminative criterion for the output weights [13]. Formally, the algorithm is as follows:

Step 1: Compute $z_i^s = \dfrac{\widehat{\alpha}_s \, \mathcal{N}(\mathbf{x}_i|\widehat{\mathbf{t}}_s, \widehat{C}_s)}{\sum_{r=1}^{k} \widehat{\alpha}_r \, \mathcal{N}(\mathbf{x}_i|\widehat{\mathbf{t}}_r, \widehat{C}_r)}$, for $i = 1, ..., N$, and $s = 1, ..., k$.

Step 2: Update weight estimates $\widehat{\alpha}_s$; see text for details.

Step 3: Update the centers and covariances: for $s = 1, ..., k$,

$$\widehat{\mathbf{t}}_s = \left(\sum_{i=1}^{N} \mathbf{x}_i \; z_i^s\right) \left(\sum_{i=1}^{N} z_i^s\right)^{-1},$$

$$\widehat{\mathbf{C}}_s = \left(\sum_{i=1}^{N} (\mathbf{x}_i - \widehat{\mathbf{t}}_s)(\mathbf{x}_i - \widehat{\mathbf{t}}_s)^T z_i^s\right) \left(\sum_{i=1}^{N} z_i^s\right)^{-1}.$$

Step 4: Update weights: $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{y} - \mathbf{p})$.

Step 5: Check some stopping criterion; if not met, go back to Step 1.

The steps of the algorithm require some explanations:

- Steps 1 and 3 are standard E and M steps of EM for Gaussian mixtures.
- In step 2, we can use one of two options: **(a)** $\widehat{\alpha}_s = |\beta_s| (\sum_{i=1}^{k} |\beta_i|)^{-1}$, which links the mixture weights to the output weights obtained by LR; **(b)** the standard update rule of EM for mixtures, $\widehat{\alpha}_s = (1/N) \sum_i z_i^s$, with no link between the output weights obtained by LR and the mixture weights. These two versions are called EM-LR-Link and EM-LR, respectively.
- Step 4 is a standard Newton-Raphson step for LR (see, e.g., [12]); $\mathbf{y}$ is a $N \times 1$ vector holding the class labels; $\mathbf{X}$ is the $N \times (k+1)$ so-called *design matrix*, that is, $X_{ij} = \mathcal{N}(\mathbf{x}_i|\widehat{\mathbf{t}}_j, \widehat{\mathbf{C}}_j)$, with the exception of the first row which is filled with ones; $\mathbf{p}$ is a $N \times 1$ vector, with element $p_i = \text{logistic}(g(\mathbf{x}_i))(1 - g(\mathbf{x}_i))$; finally, following the approach in [14], we use $\mathbf{W} = 0.25 \, \mathbf{I}_N$ (where $\mathbf{I}_N$ denotes the $N \times N$ identity matrix).
- The stopping criterion can be a maximum number of iterations, a target error rate on the training set, or some other rule.

Initialization is done by randomly placing the centers, taking identity covariances, $\alpha_s = 1/k$, and $\boldsymbol{\beta} = 0$. We have also considered a third version of the algorithm, named EM-LR-Link-K, which starts with $k = 1$, and increases this

number along the algorithm, after every Δ iterations, until the maximum value k_M is reached. This version, by starting with only one Gaussian, is much less sensitive to initialization, a critical aspect of most RBFN training algorithms.

3 Results

We compare our algorithms with some two-stage algorithms: K-means followed by LS (KMeans+LS); EM estimation of centers and covariances followed by LS weight estimation (EM+LS), as proposed in [1]; EM estimation of centers and covariances followed by LR weight estimation (EM+LR). Using the same network topology, the K-means and EM algorithms start with the same initial conditions. We evaluate generalization ability of the obtained networks (in terms of classification error rates) using test sets.

3.1 Synthetic Data

We generate a bivariate ($n = 2$) data set with 100 training and 200 test points. Fig. 1 shows the test set results, after 15 iterations, for a network with $k = 6$. The EM-LR-Link algorithm leads to zero errors, while KMeans+LS produces 21 errors. Notice the ellipsoidal probability level curves obtained by our algorithm, due to the use of full covariances. The 0.5 level curve is the decision boundary in the case of equal misclassification loss.

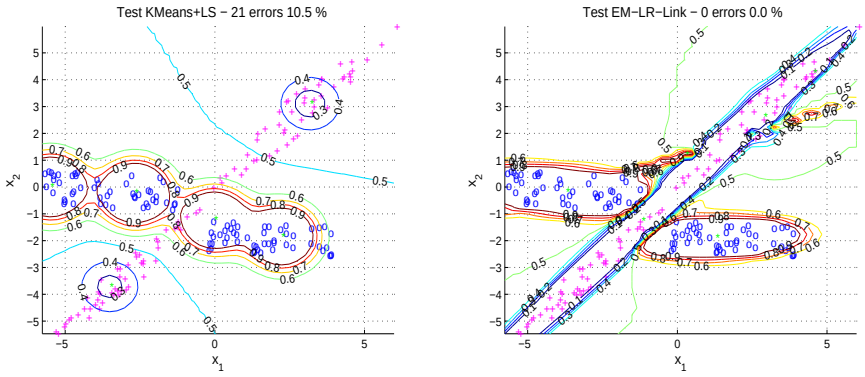

Fig. 1: Classification results obtained by KMeans+LS and our EM-LR-Link algorithm, on synthetic data. Note: the figures are best seen on color.

3.2 Benchmark Datasets

Results on the well-known Ripley dataset (for which the Bayes error rate is known to be 8%), with $k = 7$, are shown in Fig. 2. The two-stage algorithms (EM+LS and EM+LR) achieve the same error rate as EM-LR (11.1 %), while EM-LR-Link performs better (10.1 %). Fig. 3 (left plot) shows the evolution of

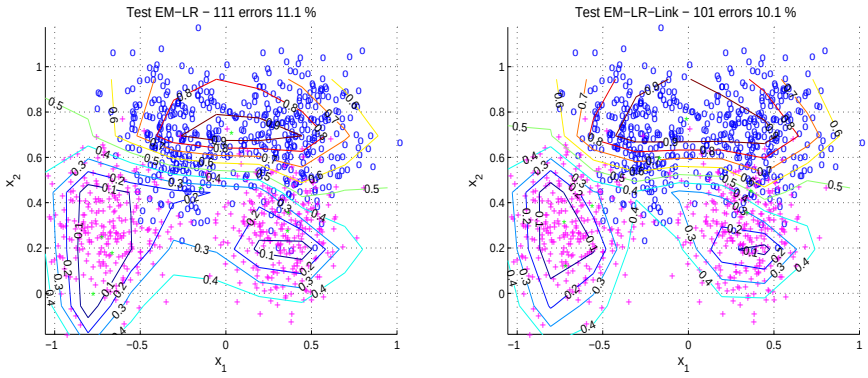

Fig. 2: Comparison of EM-LR and EM-LR-Link, on the Ripley dataset.

the train and test error rates of the EM-LR-Link-K algorithm, for the Ripley dataset, showing that it achieves a test error rate of 8.7%, close the Bayes optimal of 8%. The algorithm was run with $\Delta = 2$ and $k_M = 15$. Also in Fig. 3 (right hand side), we show a similar plot for the Crabs dataset, which contains 80 training and 120 test points, in 5 dimensions ($n = 5$); we have used $\Delta = 3$ and $k_M = 7$.

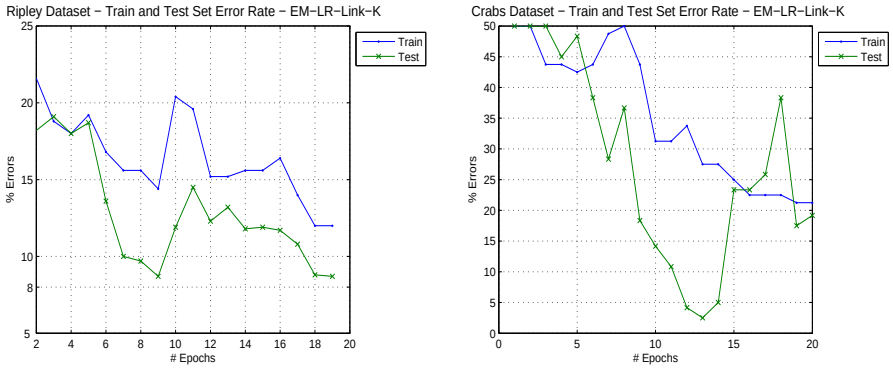

Fig. 3: Train and test set error rates, for the Ripley and Crabs datasets, along the iterations (epochs) of EM-LR-Link-K.

4 Conclusions

We have presented a global training algorithm for RBFN, combining discriminative and generative criteria. We have proposed three algorithms, integrating EM with logistic regression. One of these algorithms is incremental in the number of hidden neurons, thus much less sensitive to the initial conditions. A compar-

ison between the proposed algorithms and the conventional two-stage training schemes was carried out for binary classification on synthetic and real data.

We concluded that the proposed algorithms achieve better performance than the conventional two-stage approaches. In particular, the incremental version, named EM-LR-Link-K, produces the best results on the considered datasets and has the important advantage of being largely insensitive to initialization.

Future work includes the development of adaptive criteria for incrementing the number of hidden neurons, as well as criteria for deciding when to stop adding more neurons, to avoid over-fitting.

References

[1] Man-Wai Mak and Sun-Yuan 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.

[2] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, N.J.: Prentice Hall, 2nd edition, 1999.

[3] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley & Sons, 2nd edition, 2001.

[4] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294, 1989.

[5] M. Orr. Combing regression trees with RBFN. *International Journal of Neural Systems*, 10:453–465, 2000.

[6] M. Kubat. Decision Trees can Initialize Radial-Basis Function Networks. *IEEE Transactions on Neural Networks*, 9(5):813–821, 1998.

[7] M. Orr. An EM-algorithm for regularised RBFN. In *International Conference on Neural Networks and Brain*, Beijing, China, 1998.

[8] M. Orr. Recent advances in radial basis function network. Technical report, Institute for Adaptative and Neural Computation, Division of Informatics, Edinburg, Scotland, 1999.

[9] L. Xu. RBF nets, mixture experts, and Bayesian ying-yang learning. *Neurocomputing*, 19:223–257, 1998.

[10] M. Titsias and A. Likas. Shared kernel models for class conditional density estimation. *IEEE Trans. on Neural Networks*, 12(5):987–997, September 2001.

[11] 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.

[12] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2nd edition, 2001.

[13] A. Ng and M. Jordan. On discriminative vs. generative classifiers. In *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[14] D. Böhning. Multinomial logistic regression algorithm. *Annals of the Institute of Statistical Mathematics*, 44:197–200, 1992.