A Probabilistic RBF Network for Classification

M. Titsias and A. Likas
Department of Computer Science
University of Ioannina
45110 Ioannina - GREECE
e-mail: mtitsias@cs.uoi.gr, arly@cs.uoi.gr

Abstract

We present a probabilistic neural network model which is suitable for classification problems. This model constitutes an adaptation of the classical RBF network where the outputs represent the class conditional distributions. Since the network outputs correspond to probability densities functions, training process is treated as maximum likelihood problem and an Expectation-Maximization (EM) algorithm is proposed for adjusting the network parameters. Experimental results show that proposed architecture exhibits superior classification performance compared to the classical RBF network.

1 Introduction

In pattern recognition it is well-known that a convenient way to consider a classifier is on the basis of inferring the a posterior probabilities of each class. From the statistical point of view this inference can be achieved by first evaluating the class conditional densities $p(x|k)$ and the corresponding prior probabilities $P(k)$ and then making optimal decisions for new data points by combining these quantities through Bayes theorem

$$P(k|x) = \frac{p(x|k)P(k)}{\sum_{k'} p(x|k')P(k')} \tag{1}$$

and selecting the class with maximum $P(k|x)$. Consequently, the above approach is based on the evaluation of each class conditional density $p(x|k)$ which is estimated separately by considering only the data points of the corresponding class k.

In contrast to the statistical approach, classification neural network models (MLP, RBF) do not make probabilistic assumptions about the network outputs, however, we can arrange for the outputs to approximate the a posterior probabilities through appropriate normalization [1]. In what concerns RBF, this network provides in the hidden layer density estimation of all data. In this sense, the main differences of RBF with the previously described statistical approach is that kernel functions (hidden units) are shared among classes and that the training points of all classes contribute to the evaluation of all class conditional densities.

In the proposed probabilistic RBF (PRBF) network we combine characteristics of the statistical and neural approaches. In particular, we have developed an RBF neural network which provides output values corresponding to the class conditional densities $p(x|k)$. Since the network is RBF, the kernels are shared among classes and each class conditional density is evaluated using not only the corresponding class data points (as in the traditional statistical approach), but using all available data points. In order to train the PRBF network, an Expectation-Maximization (EM) algorithm has been derived, which provides a fast iterative procedure for adjusting the network parameters. We demonstrate the effectiveness of the proposed method using several data sets and provide also a comparison with the classical RBF network trained using a two-stage procedure [1, 3].

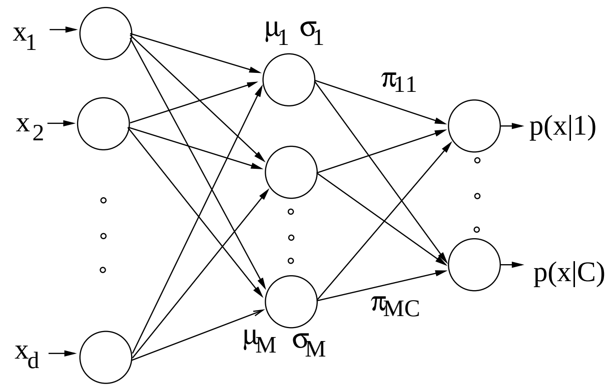

Figure 1: The architecture of the probabilistic RBF network.

2 Description of the Network

Consider a classification problem with C classes and a training set X having N supervised pairs (x^n, k^n) where $x^n \in R^d$ and k^n is an integer indicating the class of the pattern x^n. The original set X can easily be partitioned into C independent subsets X_k with N_k elements, where $k = 1, \ldots, C$, so that each subset contains only the data of the corresponding class.

As mentioned in the introduction our intention is to model the class conditional densities using an RBF network. The network architecture is displayed in Fig. 1. Typically this probabilistic RBF network has d input units and C output units (one for each class). The main difference with a classical RBF network lies on the specific functional form of the basis functions which are considered to be densities functions as well as on some constraints involving weights from the hidden to the output layer. More specifically, each basis function j ($j = 1, \ldots, M$) in the hidden layer is a Gaussian kernel of the form

$$p(x|j) = \frac{1}{\sqrt{(2\pi\sigma_j^2)^d}} \exp\{-\frac{||x - \mu_j||^2}{2\sigma_j^2}\} \tag{2}$$

where μ_j is a vector representing the center of the j kernel and σ_j^2 is the corresponding variance. This specific form of the kernel function assumes that the components of the pattern x are independent and can be represented by a common variance. Each output unit k provides density function values $p(x|k)$ for the corresponding class k in the following way:

$$p(x|k) = \sum_{j=1}^{M} \pi_{jk} p(x|j) \tag{3}$$

where the weight π_{jk} represents the prior probability that a data point has been generated from kernel j, given that it belongs to class k. These parameters satisfy the constraints:

$$\sum_{j=1}^{M} \pi_{jk} = 1 \quad and \quad \pi_{jk} \geq 0 \tag{4}$$

for every k.

Using the Bayes theorem we can compute the a posterior probability $P(j|k, x)$ that kernel j is responsible for generating pattern x given that it belongs to class k

$$P(j|k, x) = \frac{\pi_{jk} p(x|j)}{\sum_{j'} \pi_{j'k} p(x|j')}. \tag{5}$$

From the above description it is clear that the adjustable parameters of the PRBF network are the means μ_j and variances σ_j^2 of the M Gaussian kernels and also the priors π_{jk}. We denote the whole parameter vector by θ. In the next section we provide a training approach based on the EM algorithm for likelihood maximization.

3 Maximum Likelihood

Let $P(k)$ where $k = 1, \ldots, C$ denotes the prior probability of the k class. In order to use Bayes rule (1) for unlabeled input data we have to find first appropriate values for both prior probabilities and parameter vector θ (training PRBF). The whole adjustable parameter vector is $\theta' = (\theta, P(1), \ldots, P(C))$. Assuming that all data points have been independently drawn from an underlying process, we can write the log-likelihood function of the dataset X as

$$L(\theta') = \log p(X|\theta') = \sum_{n=1}^{N} \log p(x^n, k^n). \tag{6}$$

Using the relation $p(x, k) = P(k)p(x|k)$ and the fact that the dataset X consists of C independent subsets X_k, the above equation takes the form

$$L(\theta') = \sum_{k=1}^{C} N_k \log P(k) + \sum_{k=1}^{C} \sum_{n=1}^{N_k} \log p(x^n|k). \tag{7}$$

Maximization of the first term in the above equation yields $P(k) = N_k/N$ $(k = 1, \ldots, C)$, while the maximization of the second term is equivalent to training the PRBF network. Consequently, the appropriate log-likelihood function for PRBF training is given by

$$L(\theta) = \sum_{k=1}^{C} \sum_{n=1}^{N_k} \log p(x^n|k). \tag{8}$$

In order to maximize $L(\theta)$ it is possible to employ computationally intensive nonlinear optimization techniques. Nevertheless, since we seek maximum likelihood estimates, it is also possible to employ the iterative EM algorithm [2]. In the following we describe our approach to PRBF training that is based on the EM algorithm and we show that each iteration of the EM algorithm can be performed analytically leading to a fast, effective and easily implemented training scheme.

3.1 The EM algorithm

The Expectation-Maximization (EM) algorithm [2] is a general technique for maximum likelihood estimation. The algorithm assumes the existence of two data sets; the incomplete data set that consists of the actual observations and the hypothetical complete data set which contains some additional values called unobservable or hidden variables. The notion of hidden variables suggests that the problem to be solved would be straightforward if these variables were known. One iteration of the EM algorithm consists of two steps: i) the expectation step (E-step) where the expected value of the log-likelihood of the complete data set is evaluated, given the current parameter vector and the incomplete data set and ii) the maximization step (M-step) where this expected value is maximized with respect to the parameters of the model.

In order to apply the EM for maximizing (8) we have to express the complete data set; the corresponding incomplete data set is X. Similarly to EM framework for mixture models [4] the problem we have to overcome is that each data point is not followed by a label indicating the kernel which generated it. We express this missing information by introducing for each data point x^n a variable z^n which is a M-dimensional vector of one-zero values specifying the kernel that generated

x^n. If x^n was generated from kernel j, then $z_j^n = 1$, otherwise $z_j^n = 0$. Using these hidden variables the complete data set Y is defined as follows:

$$Y = \{y^1, \dots, y^N\}, \quad where \quad y^n = (x^n, k^n, z^n) \tag{9}$$

and the corresponding log-likelihood function is written in the form

$$L_C(\theta) = \sum_{n=1}^{N} \sum_{j=1}^{M} z_j^n \log\{\pi_{jk^n} p(x^n|j)\}. \tag{10}$$

At the $t+1$ iteration the current expected value of the z_j^n, given the data point x^n is equal to the a posterior probability $P^{(t)}(j|k^n, x^n)$ where t reminds us that this probability have been evaluated using the current parameters $\theta^{(t)}$. Eventually, the quantity to be maximized in the M-step is given by

$$Q(\theta; \theta^{(t)}) = \sum_{k=1}^{C} \sum_{n=1}^{N_k} \sum_{j=1}^{M} P^{(t)}(j|k, x^n)\{\log \pi_{jk} + \log p(x^n|j)\}. \tag{11}$$

It can be shown that M-step is analytically implemented. The above equation can be written as $Q = Q_1 + Q_2$ where

$$Q_1(\theta; \theta^{(t)}) = \sum_{k=1}^{C} \sum_{n=1}^{N_k} \sum_{j=1}^{M} P^{(t)}(j|k, x^n) \log \pi_{jk} \tag{12}$$

and

$$Q_2(\theta; \theta^{(t)}) = \sum_{k=1}^{C} \sum_{n=1}^{N_k} \sum_{j=1}^{M} P^{(t)}(j|k, x^n) \log p(x^n|j). \tag{13}$$

The quantity Q_1 depends solely on the parameters π_{jk}, while the quantity Q_2 depends solely on the parameters of the kernels. In order to maximize Q_1 we must take into account the constrain $\sum_{j=1}^{M} \pi_{jk} = 1$ which holds for every class k. Therefore we introduce C Lagrange multipliers λ_k and after performing some algebra we finally find that the update equation of the prior π_{jk} at the M-step is

$$\pi_{jk}^{(t+1)} = \frac{1}{N_k} \sum_{k=1}^{N_k} P^{(t)}(j|k, x^n) \tag{14}$$

for $k = 1, \dots, C$ and $j = 1, \dots, M$. Taking the derivatives of Q_2 with the respect to μ_j and σ_j^2 respectively and setting them to zero we obtain the following equations for $j = 1, \dots, M$:

$$\mu_j^{(t+1)} = \frac{\sum_{k=1}^{C} \sum_{n=1}^{N_k} P^{(t)}(j|k, x^n) x^n}{\sum_{k=1}^{C} \sum_{n=1}^{N_k} P^{(t)}(j|k, x^n)} \tag{15}$$

$$(\sigma_j^2)^{(t+1)} = \frac{1}{d} \frac{\sum_{k=1}^{C} \sum_{n=1}^{N_k} P^{(t)}(j|k, x^n) ||x^n - \mu_j^{(t+1)}||^2}{\sum_{k=1}^{C} \sum_{n=1}^{N_k} P^{(t)}(j|k, x^n)} \tag{16}$$

Starting from some initial parameter values, we perform alternatively the E-step and M-step until we reach convergence.

In the following we summarize the simple training procedure for the PRBF network:

1. Determine the number of kernels M and the initial parameter vector $\theta^{(0)}$.

2. Set $t := 0$ and compute the initial log-likelihood $L^{(0)}$ (8).

3. **Repeat**

	Number of kernels				
Algorithm	6	8	10	12	14
PRBF	11.13	10.46	10.43	10.3	10.2
RBF	25.46	23.5	23.2	22.94	22.04

Table 1: Generalization error for the clouds data set.

	Number of kernels			
Algorithm	12	18	24	30
PRBF	15.66	15.77	15.06	13.95
RBF	16.51	15.85	14.7	14.28

Table 2: Generalization error for the Satimage dataset.

(a) E-step: Compute $P^{(t)}(j|k, x^n)$ for each $k = 1, \ldots, C$, $n = 1, \ldots, N_k$ and $j = 1, \ldots, M$.

(b) M-step: Compute the new parameter values $\pi_{jk}^{(t+1)}$, $\mu_j^{(t+1)}$ and $(\sigma_j^2)^{(t+1)}$ using (14), (15) and (16) respectively.

(c) Set $t := t + 1$ and compute the new log-likelihood $L^{(t)}$.

4. **until** $|L^{(t)} - L^{(t-1)}| < \epsilon$ where ϵ determines the strictness of the convergence criterion.

4 Experimental Results and Conclusions

In this section we compare our method with the classical RBF network which has linear outputs and spherical Gaussian basis functions, described by (2) without the normalization factor. For training such a RBF network a two-stage procedure is used. In the first stage the basis functions parameters are determined by fitting a Gaussian mixture model using EM, while in the second stage the basis functions are kept fixed and the second layer weights are found by solving a set of linear equations giving rise to least squares solution. This implementation of the RBF training was found in the Netlab toolbox [3].

For the experiments we have considered data sets from ELENA database available via anonymous ftp from ftp.dice.ucl.ac.be. We have chosen one artificial dataset (Clouds) and two real datasets (Satimage and Phoneme).

For each dataset, in order to obtain an estimate of the generalization error, we have employed the K-fold cross-validation method with $K = 5$. Tables 1-3 provide the obtained results for the PRBF and RBF networks, for several values of the number of kernel functions M. These results indicate that the proposed PRBF network trained using the EM algorithm provides superior performance compared to the classical RBF network. It must also be noted that the method is fast, since in all experiments 100 EM iterations were sufficient for reaching the final solution.

In what concerns future enhancements of the method, our current work focuses on developing an approach for dynamically adjusting the number of kernels M, which constitutes the main issue in RBF network training. Our aim is to exploit recent results for adjusting the number kernels in a Gaussian mixture that have developed in the framework of pdf estimation (unsupervised learning) [5, 6].

References

[1] C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

	Number of kernels				
Algorithm	6	8	10	12	14
PRBF	22.18	21.59	21.33	20.9	21.16
RBF	24.12	24.5	24.57	24.0	24.12

Table 3: Generalization error for the Phoneme dataset.

[2] A. P. Dempster, N. M. Laird and D. B. Rubin, "Maximum Likelihood Estimation from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society B*, vol. 39, pp. 1-38, 1977.

[3] I. Nabney and C. Bishop, *Netlab: Neural Network Software*, available from http://www.ncrg.aston.ac.uk/netlab

[4] R. Redner and H. Walker, "Mixture densities, maximum likelihood and the EM algorithm", *SIAM Review*, vol. 26, no. 2, pp. 195-239, 1984.

[5] N. A. Vlassis, G. Papakonstantinou and P. Tsanakas, "Mixture Density Estimation based on Maximum Likelihood and Test Statistics", *Neural Processing Letters*, vol. 9, no. 1, 1999.

[6] N. A. Vlassis and A. Likas, "A Kurtosis-Based Dynamic Approach to Gaussian Mixture Modeling", IEEE Trans. on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 29, no. 4, pp. 393-399, 1999.