THE PROBABILISTIC RBF NETWORK WITH CONSTRAINED KERNEL SHARING

Michalis K. Titsias and Aristidis Likas

2-2001

Preprint no. 2-02/2001

Department of Computer Science
University of Ioannina
451 10 Ioannina, Greece

The Probabilistic RBF network with constrained kernel sharing

Michalis K. Titsias and Aristidis Likas

Department of Computer Science
University of Ioannina
45110 Ioannina, Greece
e-mail: {mtitsias,arly}@cs.uoi.gr

Abstract. The Probabilistic RBF (PRBF) network constitutes a classification network that employs Gaussian mixture models for class conditional density estimation [7, 8]. The particular characteristic of this model is that it allows the sharing of the Gaussian components of the mixture models among all classes in the same spirit that the hidden units of a classification RBF network feed all output units. In the present work, we present an extension to the original PRBF network that allows to impose constraints on the degree of kernel sharing among the classes. We also propose a fast and iterative training procedure (based on the EM algorithm) that automatically adjusts not only the kernel parameters but also the degree of sharing of each kernel among the classes. Experimental results on well-known classification data sets indicate that the proposed method leads to superior generalization performance compared to the original PRBF network with the same number of kernels (hidden units).

1 Introduction

The Probabilisitc RBF network (PRBF) is an adaptation of the classical RBF network where the hidden units correspond to the Gaussian kernels and each output represents the class conditional density of each class [7, 8]. From the point of view of gaussian mixture modeling, a PRBF network is considered to employ one mixture model for the conditional density of each class, with the characteristic that each gaussian kernel is allowed to contribute to the density estimation of all classes.

More specifically, $p(x|j, \theta_j)$ denotes the gaussian probability density computed by kernel (hidden unit) j (with parameter vector θ_j). For a PRBF network with K classes and M kernels, the class conditional density $p(x|C_k)$ corresponding to output k is given by

$$p(x|C_k; \pi_k, \theta) = \sum_{j=1}^{M} \pi_{jk} p(x|j; \theta_j) \qquad k = 1, \ldots, K, \tag{1}$$

where π_{jk} is the mixture coefficient (prior) representing the prior probability of component j conditioned on the class C_k. We denote with π_k the vector of

all the priors associated with the class C_k and with θ the vector of all hidden unit parameters $\{\theta_j\}$. The priors cannot be negative and for each k satisfy the constraint

$$\sum_{j=1}^{M} \pi_{jk} = 1. \tag{2}$$

In order to train the PRBF network a fast training procedure has been developed based on the EM algorithm [7].

As already stated the most important characteristic of the PRBF network is that it allows for each gaussian kernel j to contribute to the density estimation of every class C_k. In addition to adjust the parameters of each kernel during training all available training data are used. This is in contrast to the conventional application of mixture models to classification, where a separate mixture model is used for each class and is trained independently using only the data of the corresponding class.

The most common argument concerning the usefulness of kernel sharing is that it can be beneficial in cases of highly overlapped classes since a kernel is able to contribute simultaneously to the modeling of different classes. However, in real world problems we are not aware of the kind of overlapping between the classes and the employment of the above models might in some cases lead to poor data representations from the classification perspective.

Therefore, in many cases it would be beneficial if we could constrain the degree with which a kernel contributes to the mixture model of a specific class. In this spirit, we present in this work *constrained PRBF* models and also present training algorithms to automatically adjust the degree of kernel sharing. We next present the formulation for this type of model and provide experimental results where for fixed number of kernels the solution with the best classification performance is obtained by constrained shared kernel models.

2 The constrained PRBF network

Suppose we are given a classification problem with K classes expressed through a data set X with N pairs (x^n, k^n) where $x^n \in R^d$ and k^n denotes the class of pattern x^n.

As stated in the introduction, the model described by (1) can be beneficial in cases of highly overlapped classes and especially in such cases constistute a way of reducing (compared to the separate mixture models) the required number of kernels for representing efficiently the data. On the other hand, if the classes exhibit little overlap then the modeling (1) may fail to provide with a qualified data representation from the classification perspective [8]. The proposed method attempts to resolve the above problem by automatically adjusting during training the degree of kernel sharing among classes.

To begin with, we define the following functions[1] assuming a fixed number M of kernels:

$$\phi(x; C_k, r_k, \pi_k, \theta) = \sum_{j=1}^{M} r_{jk}\pi_{jk}p(x|j;\theta_j) \quad k = 1, \ldots, K, \tag{3}$$

which actually are those defined by (1) with special constraint parameters r_{jk} incorporated in the linear sum. These parameters cannot be negative and for each j satisfy the constraint:

$$\sum_{k=1}^{K} r_{jk} = 1. \tag{4}$$

The role of each parameter r_{jk} is to specify the degree at which the kernel j is allowed to be used for modeling class C_k, that is to represent data of this class. In the definition of the ϕ functions we can observe that the incorporation of the r variables adds competition between classes concerning the use of the kernels, which is essentially due to the constraint (4). This competition will become more clear later when we will introduce an objective function for learning the parameters (Θ, r), where Θ denotes the whole parameter vector containing both kernel parameters and priors, while r denotes the vector of parameters r_{jk}. The values of the priors π_{jk} satisfy by definition

$$\sum_{j=1}^{M} \pi_{jk} = 1 \;\; and \; if \; r_{jk} = 0 \; then \; \pi_{jk} = 0, \tag{5}$$

for each k. Therefore for every class C_k there must be at least one $r_{jk} > 0$. Thus, we exclude the case where a function ϕ and, consequently, the corresponding class conditional density is zero.

The functions ϕ in general do not constitute densities with respect to x due to the fact that $\int \phi(x; C_k, r_k, \pi_k, \theta)dx \leq 1$, unless the constraints r_{jk} are assigned zero-one values[2]. However, it holds that $\phi(x; C_k, r_k, \pi_k, \theta) \geq 0$ and $\int \phi(x; C_k, r_k, \pi_k, \theta)dx > 0$ which is due to (5).

In order to exploit the ϕ functions for learning the model parameters (Θ, r) it is necessary to treat them as if they were class conditional probability densities. Such a learning procedure would actually realize the competition between classes implied in the definition of the ϕ functions as discussed previously. If the ϕ functions are treated in a way analogous to the corresponding class densities, we can introduce an objective function analogous to the log-likelihood function as follows:

$$L(\Theta, r) = \sum_{k=1}^{K} \sum_{x \in X_k} \log \phi(x; C_k, r_k, \pi_k, \theta) \tag{6}$$

[1] These functions do not represent the class density models to be constructed, but we use them in order to find suitable parameters for the real class density models (which are given by (1)).

[2] In this special case each function $\phi(x; C_k, r_k, \pi_k, \theta)$ is identical to the corresponding $p(x|C_k; \pi_k, \theta)$.

where X_k denotes the available data of class C_k. Through the maximization of the above function we adjust the values of the r variables (actually the degree kernel sharing) and this automatically influences the solution for the class density models parameter Θ. Intuitively, we expect that the adjustment of the r variables will be based on a strong competition among classes concerning the use of the kernels. This issue will become more obvious in the next section where we present the update parameter equations of an Expectation-Maximization (EM) algorithm for the maximization of the objective function (6).

3 Training using the EM algorithm

We can use the EM algorithm [3] for maximizing the objective function (6). Although the EM algorithm is employed for log likelihood or log posterior maximization, the fact that our objective function in its general form does not correspond to any of the above cases does not constitute a problem. It can be proved that the basic EM property concerning the guaranteed monotone increase of the objective function still holds in our case[3] (6).

The EM algorithm starts from an initial parameter point $(\Theta^{(0)}, r^{(0)})$ and proceeds iteratively performing two steps: i) the E-step, where the expected value of the complete data log likelihood function (denoted by Q) is estimated and ii) the M-step where the Q function is maximized with respect to the parameters. In our case the Q function evaluated at the E-step of the $t+1$ EM iteration is:

$$Q(\Theta, r; \Theta^{(t)}, r^{(t)}) = \sum_{k=1}^{K} \sum_{x \in X_k} \sum_{j=1}^{M} \Phi_j(x; C_k, r_k^{(t)}, \pi_k^{(t)}, \theta^{(t)}) \log\{r_{jk} \pi_{jk} p(x|j; \theta_j)\} \tag{7}$$

where

$$\Phi_j(x; C_k, r_k^{(t)}, \pi_k^{(t)}, \theta^{(t)}) = \frac{r_{jk}^{(t)} \pi_{jk}^{(t)} p(x|j; \theta_j^{(t)})}{\sum_{i=1}^{M} r_{ik}^{(t)} \pi_{ik}^{(t)} p(x|i; \theta_i^{(t)})}. \tag{8}$$

The algorithm at the M-step maximizes the above function with respect the parameter vector (Θ, r). If we assume that the mixture components are Gaussians of the general form

$$p(x|j; \mu_j, \Sigma_j) = \frac{1}{(2\pi)^{d/2} |\Sigma_j|^{1/2}} \exp\left\{-\frac{1}{2}(x - \mu_j)^T \Sigma_j^{-1} (x - \mu_j)\right\} \tag{9}$$

where μ_j is a d-dimensional vector representing the center of kernel j, while Σ_j represents the corresponding $d \times d$ covariance matrix. Then the M-step of the EM can be performed analytically for the whole parameter vector (Θ, r). More specifically the update equations for each kernel j are the following:

$$\mu_j^{(t+1)} = \frac{\sum_{k=1}^{K} \sum_{x \in X_k} \Phi_j(x; C_k, r_k^{(t)}, \pi_k^{(t)}, \theta^{(t)}) x}{\sum_{k=1}^{K} \sum_{x \in X_k} \Phi_j(x; C_k, r_k^{(t)}, \pi_k^{(t)}, \theta^{(t)})}, \tag{10}$$

[3] The proof is omitted due to space limitations

$$\Sigma_j^{(t+1)} = \frac{\sum_{k=1}^{K} \sum_{x \in X_k} \Phi_j(x; C_k, r_k^{(t)}, \pi_k^{(t)}, \theta^{(t)})(x - \mu_j^{(t+1)})(x - \mu_j^{(t+1)})^T}{\sum_{k=1}^{K} \sum_{x \in X_k} \Phi_j(x; C_k, r_k^{(t)}, \pi_k^{(t)}, \theta^{(t)})}, \quad (11)$$

$$\pi_{jk}^{(t+1)} = \frac{1}{|X_k|} \sum_{x \in X_k} \Phi_j(x; C_k, r_k^{(t)}, \pi_k^{(t)}, \theta^{(t)}), \qquad k = 1, \ldots, K \quad (12)$$

$$r_{jk}^{(t+1)} = \frac{\sum_{x \in X_k} \Phi_j(x; C_k, r_k^{(t)}, \pi_k^{(t)}, \theta^{(t)})}{\sum_{i=1}^{K} \sum_{x \in X_i} \Phi_j(x; C_i, r_i^{(t)}, \pi_i^{(t)}, \theta^{(t)})}, \qquad k = 1, \ldots, K \quad (13)$$

Using the equation (12), equation (13) can be written as

$$r_{jk}^{(t+1)} = \frac{\pi_{jk}^{(t+1)} |X_k|}{\sum_{i=1}^{K} \pi_{ji}^{(t+1)} |X_i|} \quad (14)$$

which explains how the r variables are adjusted at each EM iteration with respect to the newly estimated prior values. If we assume that the classes have nearly the same number of available training points, then during training each class C_k is constrained to use a kernel j based on the ratio of the corresponding prior value π_{jk} over the sum of the rest of the priors associated with the same kernel j.

The EM algorithm performs iterations (eq. (10 - 13)) until convergence to some locally optimal parameter point (Θ^*, r^*). Then the estimated class densities are given by (1) with parameters Θ^*.

4 Experimental Results and Conclusions

For the experiments we have considered two well-known classification data sets (Satimage and Phoneme) from ELENA database [2] and one data set (Pima indians) from the UCI repository [1]. For each dataset, in order to obtain an estimate of the generalization error, we have employed the K-fold cross-validation method with $K = 5$.

	12 kernels		18 kernels		24 kernels	
Algorithm	error	std	error	std	error	std
Constrained PRBF	12.35	0.39	11.85	0.61	11.29	0.53
PRBF	13.23	0.56	12.28	0.79	11.52	0.75

Table 1. Generalization error for the Satimage data set.

Tables 1-3 display the obtained results in terms of average value and standard deviation of the generalization error for several values of the number of kernel functions M using i) the original method for PRBF training [7, 8] and ii) the proposed constrained method for PRBF (in the tables Con. PRBF) training.

	10 kernels		12 kernels		14 kernels	
Algorithm	error	std	error	std	error	std
Constrained PRBF	18.05	0.22	17.40	0.95	15.74	0.90
PRBF	20.62	0.75	20.03	0.75	20.98	1.04

Table 2. Generalization error for the Phoneme dataset.

	10 kernels		12 kernels		14 kernels	
Algorithm	error	std	error	std	error	std
Constrained PRBF	27.47	2.37	27.73	3.26	25.52	1.99
PRBF	29.95	3.06	28.12	2.21	28.25	1.97

Table 3. Generalization error for the Pima Indians dataset.

These results indicate that the proposed technique provides in general superior performance results compared to the PRBF network with full kernel sharing.

In what concerns future enhancements of the method, current work focuses on developing Bayesian techniques for PRBF training that require the formulation of a suitable prior function (over the mixing coefficients π_{jk}) that will favour more or less kernel sharing.

References

1. C. L. Blake and C.J. Merz, "UCI repository of machine learning databases", University of California, Irvine, Dept. of Information an Computer Sciences, 1998.
2. Datasets and technical reports available via anonymous ftp from: ftp.dice.ucl.ac.be/pub/neural-nets/ELENA/databases.
3. 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.
4. G. J. McLachlan and K. Basford, *Mixture models: Inference and applications to clustering*. Wiley, 1988.
5. G. J. McLachlan and T. Krishnan, *The EM algorithm and Extensions*. Marcel Dekker, 1997.
6. R. Redner and H. Walker, "Mixture densities, maximum likelihood and the EM algorithm", *SIAM Review*, vol. 26, no. 2, pp. 195-239, 1984.
7. M. Titsias, A. Likas. "A Probabilistic RBF network for Classification", *Proc. of International Joint Conference on Neural Networks*, Komo, Italy, July 2000.
8. M. Titsias and A. Likas "Shared Kernel Models for Class Conditional Density Estimation", *IEEE Trans. on Neural Networks*, accepted with revision.