SHARED KERNEL MODELS FOR CLASS CONDITIONAL DENSITY ESTIMATION

M. Titsias and A. Likas

12-2000

Preprint no. 12-00/2000

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

Shared Kernel Models for Class Conditional Density Estimation

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

Abstract

We present probabilistic models which are suitable for class conditional density estimation and can be regarded as shared kernel models where sharing means that each kernel may contribute to the estimation of the conditional densities of all classes. We first propose a model that constitutes an adaptation of the classical RBF network (with full sharing of kernels among classes) where the outputs represent class conditional densities. In the opposite direction is the approach of separate mixtures model where the density of each class is estimated using a separate mixture density (no sharing of kernels among classes). We present a general model that allows for the expression of intermediate cases where the degree of kernel sharing can be specified through an extra model parameter. This general model encompasses both above mentioned models as special cases. In all proposed models the training process is treated as a maximum likelihood problem and EM algorithms have been derived for adjusting the model parameters.

1 Introduction

Probability density estimation constitutes an unsupervised method that attempts to model the underlying density function from which a given set of unlabeled data have been generated. An important application of density estimation is that it can be utilized for solving classification problems. A technique for constructing such classifiers is based on the separate estimation of the conditional density $p(x|C_k)$ of each class C_k [3], which means that each density estimation is carried out considering only the patterns of the corresponding class. To classify a new pattern x, the conditional densities are combined with prior probabilities $P(C_k)$ through Bayes' theorem and provide the posterior probabilities $P(C_k|x)$:

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

A density estimation approach that has been extensively used in statistical pattern recognition is based on *mixture density models* [6, 13]. For such models efficient training procedures have been developed based on the EM algorithm [2]. In classification problems separate mixture models are employed to estimate the class conditional densities. Throughout this paper we

will refer to that method as separate mixtures. Nevertheless, we argue that more general models for conditional density estimation can be derived in terms of shared kernel functions where the class conditional densities are represented by a set of kernels which may contribute to the estimation of the conditional densities of all classes. This is analogous to kernel sharing in a typical RBF network.

In this paper, we first propose a model which comprises a special case of the RBF neural network in which the basis functions are taken to be probability densities and the second layer weights are constrained to represent prior probabilities. In this way, the outputs of the RBF represent class conditional densities. This model is discussed in [1] where the basis functions of the network are considered as a common pool of kernels that represent all the class conditional densities. The discussion in [1] aims at showing how the activation functions and the second layer weights of an RBF could be defined so that the outputs to be precisely interpreted as posterior probabilities of class membership. In our case, as mentioned above, we consider an RBF model whose outputs directly represent conditional density functions. This interpretation of the outputs has given the opportunity to treat RBF training as a maximum likelihood problem and derive an one-stage EM algorithm for adjusting the model parameters. This approach seems to be more sophisticated than the unsupervised learning techniques typically used for finding the basis function parameters [1]. Because of the similarity with RBF network we call this model probabilistic RBF (PRBF) [12]. The PRBF model is presented in Section 2.

Moreover, we have further extended the PRBF model and developed a more general one, called λPRBF, that allows to express intermediate models between PRBF and separate mixtures. This model is derived from PRBF by introducing a special parameter (denoted by λ) which adds constraints to the model parameters in order to adjust kernel sharing among classes. As discussed in detail in Section 3, the role of parameter λ is to control the contribution of each kernel to the density estimation of each class. For this model we have also developed an EM algorithm for the adjustment of its parameters.

In Section 4 we demonstrate the effectiveness of the proposed methods using both artificial and real data sets and provide comparative results with other methods. Finally, Section 5 contains conclusions and research directions for future enhancements.

2 The Probabilistic RBF Model

Consider a classification problem with K classes and a training set $X = \{(x^n, k^n),\ n = 1, \ldots, N\}$ where x^n is a d-dimensional pattern and k^n is an integer in the range $(1, K)$ indicating the class of the pattern x^n. The original set X can be easily partitioned into K

independent subsets X_k so that each subset contains only the data of the corresponding class. Let N_k denote the number of patterns of class C_k, ie. $N_k = |X_k|$.

Assume that we have a number of M kernel functions, which are probability densities, and we would like to utilize them for estimating the conditional densities of all classes by considering the kernels as a common pool. Thus, each class conditional density function $p(x|C_k)$ is modeled as

$$p(x|C_k) = \sum_{j=1}^{M} \pi_{jk} p(x|j), \qquad k = 1, \ldots, K \tag{2}$$

where $p(x|j)$ denotes the kernel function j, while the mixing coefficient π_{jk} represents the prior probability of the pattern x having been generated from kernel j, given that it belongs to class C_k. The priors take positive values and satisfy the following constraint:

$$\sum_{j=1}^{M} \pi_{jk} = 1, \qquad k = 1, \ldots, K. \tag{3}$$

We will find it useful to introduce the posterior probabilities expressing our posterior belief that kernel j generated a pattern x given its class C_k. This probability is obtained using the Bayes' theorem

$$P(j|C_k, x) = \frac{\pi_{jk} p(x|j)}{\sum_{j'} \pi_{j'k} p(x|j')}. \tag{4}$$

Obviously, the posterior probabilities sum to unity

$$\sum_{j=1}^{M} P(j|C_k, x) = 1. \tag{5}$$

In the following, we assume that the kernel functions are Gaussians of the general form

$$p(x|j) = \frac{1}{(2\pi)^{d/2} |\Sigma_j|^{1/2}} \exp\left\{ -\frac{1}{2} (x - \mu_j)^T \Sigma_j^{-1} (x - \mu_j) \right\} \tag{6}$$

where $\mu_j \in R^d$ is a vector representing the center of kernel j, while Σ_j represents the corresponding $d \times d$ covariance matrix. The whole adjustable parameter vector of the model consists of the priors and the kernel parameters (means and covariances) and we denote it by θ.

It is apparent that the PRBF model (Fig. 1) is a special case of the radial basis function network where the outputs correspond to probability density functions and the second layer weights are constrained to represented prior probabilities. Furthermore, the separate mixtures model can be derived as a special case of PRBF. This is illustrated in Fig. 2. The PRBF kernels are partitioned into K disjoint groups with each group corresponding to a specific class. In this sense, each kernel j is associated with only one class $C(j)$ and the separate mixtures model is obtained by setting all the prior probabilities of a kernel equal to zero, except for the prior corresponding to class $C(j)$.

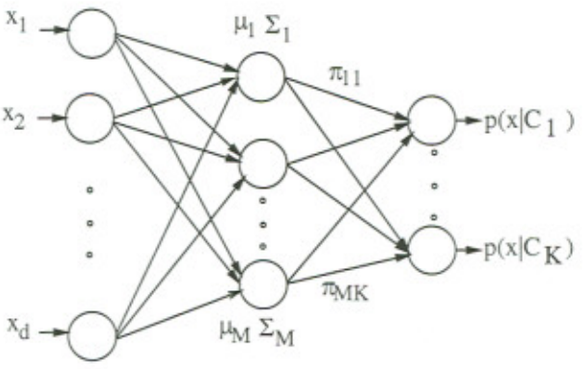


Figure 1: The architecture of the probabilistic RBF network.

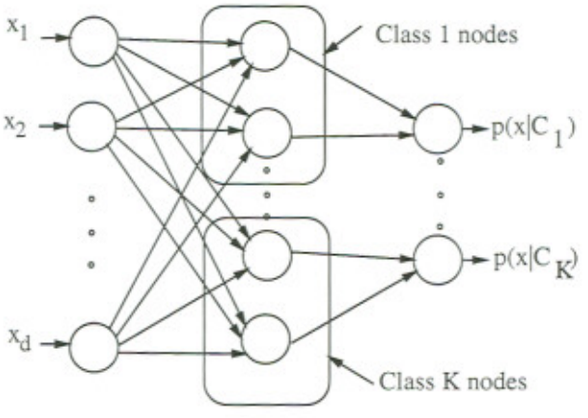


Figure 2: The separate mixtures model as a special case of the probabilistic RBF network.

2.1 Derivation of the Log-likelihood Function

Let $P(C_k)$, $k = 1, \ldots, K$ be the prior probabilities of the classes. In order to use Bayes' theorem (1) for unlabeled input data first we have to specify appropriate values for both class priors and the parameter vector θ. In our case, the maximum likelihood procedure is proven to be directly applicable. Assuming that all data have been independently drawn from an underlying process, we write the likelihood function in the form

$$p(X|\theta, P(C_1), \ldots, P(C_K)) = \prod_{n=1}^{N} p(x^n, C_{k^n}) \tag{7}$$

from which we obtain the log-likelihood function

$$L(\theta, P(C_1), \ldots, P(C_K)) = \sum_{n=1}^{N} \log p(x^n, C_{k^n}). \tag{8}$$

Now, using that $p(x, C_k) = P(C_k)p(x|C_k)$ and also the fact that the data set X consists of K independent subsets with N_k elements each, the above quantity takes the form

$$L(\theta, P(C_1), \ldots, P(C_K)) = \sum_{k=1}^{K} N_k \log P(C_k) + \sum_{k=1}^{K} \sum_{n=1}^{N_k} \log p(x^n|C_k). \tag{9}$$

Apparently, the two terms above can be maximized separately as they do not contain common parameters. Maximization of the first term yields

$$P(C_k) = \frac{N_k}{N}, \quad k = 1, \ldots, K \tag{10}$$

while the maximization of the second term is equivalent to PRBF training. Consequently, the log-likelihood function suitable for the training of the PRBF network is given by

$$L(\theta) = \sum_{k=1}^{K} \sum_{n=1}^{N_k} \log p(x^n|C_k). \tag{11}$$

To maximize $L(\theta)$ it is possible to employ nonlinear optimization techniques, however, it would be desirable to show that the iterative EM algorithm is applicable in this case. In the following we describe our approach to maximization of the above likelihood using the EM algorithm and we show that in the case of Gaussian kernels each iteration of the EM algorithm is performed analytically.

2.2 Applying EM for Training the PRBF Network

The EM algorithm [2] is defined as a very general technique for maximum likelihood estimation. The algorithm is applicable in caces where we seek maximum likelihood estimates in the

presence of unobserved variables. Several extensions and also many applications of the EM are presented in [7]. Before presenting our EM approach for training PRBF, we will briefly review the basic properties of the EM algorithm.

Assume that we have a set X of observed data, called incomplete data, and a set of unobserved variables Z which along with the observed data constitute the complete data $Y = (X, Z)$. Furthermore assume that $p(X|\theta)$ and $p(X, Z|\theta)$ are the probability densities of the incomplete and complete data, respectively, parameterized on θ. It follows that

$$p(X|\theta) = \int p(X, Z|\theta) dZ. \tag{12}$$

The EM algorithm approaches the problem of maximizing the incomplete data log-likelihood function $L(\theta) = \log p(X|\theta)$ indirectly, in terms of the complete data log-likelihood function $L_c(\theta) = \log p(X, Z|\theta)$. More specifically, the EM starts from a initial parameter guess and proceeds iteratively performing alternatively two steps: the E-step in which the algorithm calculates the expected value of the complete data log-likelihood function (with respect to the unobserved variables) given the current parameter vector $\theta^{(t)}$ and the incomplete data X

$$Q(\theta|\theta^{(t)}) = E\{L_c(\theta)|X, \theta^{(t)}\} \tag{13}$$

and the M-step, where the old parameter vector $\theta^{(t)}$ is replaced by $\theta^{(t+1)}$ obtained by maximizing $Q(\theta|\theta^{(t)})$.

In order to apply the EM algorithm to maximize (11) we must first express the unobserved variables. Similarly to the EM framework for mixture models [10], the problem we have to overcome is that each pattern is not followed by a label indicating the kernel responsible for having generated it. To express this missing information we introduce for each pattern x^n a variable z^n which is a M-dimensional vector indicating the kernel that generated x^n. More specifically, if x^n was generated from kernel j, then $z_j^n = 1$, otherwise $z_j^n = 0$. The set of the unobserved variables is $Z = \{z^n,\ n = 1, \ldots, N\}$, while the complete data set is $Y = \{(x^n, k^n, z^n),\ n = 1, \ldots, N\}$. The log-likelihood function of the complete data is given by

$$L_c(\theta) = \sum_{k=1}^{K} \sum_{n=1}^{N_k} \sum_{j=1}^{M} z_j^n \log \pi_{jk} p(x^n|j). \tag{14}$$

At iteration $t+1$ the expected value of the z_j^n (given x^n) is equal to the posterior probability $P^{(t)}(j|C_{k^n}, x^n)$, where t denotes that this probability has been computed using the current parameters $\theta^{(t)}$. It follows that the function Q takes the from

$$Q(\theta|\theta^{(t)}) = \sum_{k=1}^{K} \sum_{n=1}^{N_k} \sum_{j=1}^{M} P^{(t)}(j|C_k, x^n) \left\{\log \pi_{jk} + \log p(x^n|j)\right\}. \tag{15}$$

It can be shown that the maximization of Q can be carried out analytically. If we write the function Q as $Q = Q_1 + Q_2$ where

$$Q_1(\theta|\theta^{(t)}) = \sum_{k=1}^{K}\sum_{n=1}^{N_k}\sum_{j=1}^{M} P^{(t)}(j|C_k, x^n)\log \pi_{jk} \tag{16}$$

and

$$Q_2(\theta|\theta^{(t)}) = \sum_{k=1}^{K}\sum_{n=1}^{N_k}\sum_{j=1}^{M} P^{(t)}(j|C_k, x^n)\log p(x^n|j) \tag{17}$$

then we can maximize separately the above quantities since they do not contain common parameters. In order to maximize Q_1 we must take account of the constraints involving priors (3). Therefore, we introduce K Lagrange multipliers and the quantity Q_1^L to be maximized takes the form

$$Q_1^L(\theta|\theta^{(t)}) = Q_1(\theta|\theta^{(t)}) - \sum_{k=1}^{K}\lambda_k\left(\sum_{j=1}^{M}\pi_{jk} - 1\right). \tag{18}$$

Expressing the derivatives of Q_1^L with respect to priors π_{jk}, we easily obtain $\lambda_k = N_k$, $k = 1, \ldots, K$. Also the differentiation of Q_2 with respect to the kernel parameters leads to the following update equations:

$$\mu_j^{(t+1)} = \frac{\sum_{k=1}^{K}\sum_{n=1}^{N_k} P^{(t)}(j|C_k, x^n)x^n}{\sum_{k=1}^{K}\sum_{n=1}^{N_k} P^{(t)}(j|C_k, x^n)} \tag{19}$$

$$\Sigma_j^{(t+1)} = \frac{\sum_{k=1}^{K}\sum_{n=1}^{N_k} P^{(t)}(j|C_k, x^n)(x^n - \mu_j^{(t+1)})(x^n - \mu_j^{(t+1)})^T}{\sum_{k=1}^{K}\sum_{n=1}^{N_k} P^{(t)}(j|C_k, x^n)} \tag{20}$$

$$\pi_{jk}^{(t+1)} = \frac{1}{N_k}\sum_{k=1}^{N_k} P^{(t)}(j|C_k, x^n) \quad k = 1, \ldots, K \tag{21}$$

where $j = 1, \ldots, M$. Starting from an initial parameter vector, we first calculate the posterior probabilities and then we update the parameter values using the above equations (19)-(21). We perform these steps alternatively until convergence.

In the following, we summarize the training algorithm for the PRBF network:

1. Specify the number of kernels M and the initial parameter values.
2. E-step: For each training point $(x^n, k^n) \in X$ compute the posterior probabilities $P^{(t)}(j|C_{k^n}, x^n)$, $j = 1, \ldots, M$, from (4) using the current parameters $\theta^{(t)}$.
3. M-step: Find the new parameter vector $\theta^{(t+1)}$ from equations (19), (20) and (21) respectively.
4. Iterate going to step 2 until a local maximum of the log-likelihood (11) is reached.

When an RBF neural network is employed for classification problems, the parameters of basis functions are typically specified by unsupervised techniques such as the K-means clustering algorithm or Gaussian mixture modeling with EM. After the basis function parameters have been computed, the second layer weights are optimized rapidly using supervised learning. However, the determination of the basis functions parameters using unsupervised learning techniques cannot be regarded as an efficient approach, since it does not make use of class labels and therefore it might lead to undesirable situations. For example, after unsupervised training, it is possible for a kernel to represent data of several classes, even if these classes are linearly discriminated and given that the number of kernels is large enough to sufficiently represent the data. As it is shown in the next section, the proposed EM algorithm for PRBF training generally does not adjust the kernel parameters similarly to unsupervised learning methods, but there is an active competition among classes concerning kernel allocation.

2.3 Adjustment of Kernel Parameters in PRBF Training

According to equations (19) and (20) the means and covariances of each kernel are updated using data from all classes. This may cause confusion concerning the operation characteristics of the algorithm. At first glance, the algorithm seems to adjust the kernel parameters estimating the distribution of all data, that is similar to unsupervised techniques. However, as it is shown next by writing the equations (19)-(20) in a suitable form, the algorithm works quite differently giving emphasis to the classification problem.

The posterior probability that a pattern x belongs to class C_k, given that it has been generated from kernel j, can be expressed as

$$P(C_k|j) = \frac{\pi_{jk} P(C_k)}{\sum_{k'=1}^{K} \pi_{jk'} P(C_{k'})} \tag{22}$$

and is independent of x. Apparently $\sum_{k=1}^{K} P(C_k|j) = 1$. The probability $P(C_k|j)$ can also be interpreted as expressing the degree at which kernel j represents data of class C_k.

Let now assume that the algorithm is at iteration $t+1$ and the the E-step has been completed. We introduce the variables $\mu_{jk}^{(t+1)}$ and $\Sigma_{jk}^{(t+1)}$ $(j = 1, \ldots, M,\ k = 1, \ldots, K)$, which represent means and covariances matrices respectively as follows:

$$\mu_{jk}^{(t+1)} = \frac{\sum_{n=1}^{N_k} P^{(t)}(j|C_k, x^n) x^n}{\sum_{n=1}^{N_k} P^{(t)}(j|C_k, x^n)} \tag{23}$$

$$\Sigma_{jk}^{(t+1)} = \frac{\sum_{n=1}^{N_k} P^{(t)}(j|C_k, x^n)(x^n - \mu_{jk}^{(t+1)})(x^n - \mu_{jk}^{(t+1)})^T}{\sum_{n=1}^{N_k} P^{(t)}(j|C_k, x^n)}. \tag{24}$$

Using these notations, we can express the EM update equations in an appropriate form. If we let the parameter w_{jk} denote either the mean μ_{jk} or the covariance matrix Σ_{jk}, and, similarly,

the parameter w_j denote either μ_j or Σ_j, then we can write that

$$w_j^{(t+1)} = \frac{\sum_{k=1}^{K} \left\{ \sum_{n=1}^{N_k} P^{(t)}(j|C_k, x^n) \right\} w_{jk}^{(t+1)}}{\sum_{k=1}^{K} \sum_{n=1}^{N_k} P^{(t)}(j|C_k, x^n)}. \tag{25}$$

Using (21), (22) and also (10) we finally find that

$$w_j^{(t+1)} = \frac{\sum_{k=1}^{K} \left\{ \pi_{jk}^{(t+1)} P(C_k) \right\} w_{jk}^{(t+1)}}{\sum_{k=1}^{K} \pi_{jk}^{(t+1)} P(C_k)} = \sum_{k=1}^{K} P^{(t+1)}(C_k|j) w_{jk}^{(t+1)}. \tag{26}$$

The above equation indicates that the parameters of kernel j at iteration $t+1$ constitute the expected values of the variables $\mu_{jk}^{(t+1)}$ and $\Sigma_{jk}^{(t+1)}$, $k = 1, \ldots, K$, with the corresponding class probabilities given by (22). Consequently, the new parameter values w_j of the kernel j obtained from an EM iteration during PRBF training can be interpreted as the mean values of the corresponding parameters w_{jk} that are obtained from K underlying EM procedures. Each EM procedure corresponds to a specific class C_k and updates the parameters w_{jk} using only data of class C_k. This suggests that each class C_k competes to 'allocate' a kernel j (ie. setting w_j closer to w_{jk}) and this competition is expressed in terms of the values $P(C_k|j)$. For example, if there exists a class C_k having high value for the probability $P(C_k|j)$, the new parameter values of kernel j will be close to those values obtained from the EM iteration considering the data of the class C_k.

In the following we illustrate through an example how the algorithm operates compared to unsupervised learning. We have created a simple synthetic two-dimensional data set that is a mixture of three Gaussian kernels. Two of the Gaussians correspond to the first class and the third to the second class (Fig. 3). We applied the EM algorithm for training PRBF (supervised training) and also the EM for density estimation ignoring class labels (unsupervised training). In both experiments, two kernels were used with common parameter initialization. As Fig. 3 indicates, the EM algorithm for training PRBF places one of the kernels in a sensitive way so as to represent all data of the first class, while the unsupervised training places the kernels so as to approximate the density of all data. A serious implication of the above remark is that the PRBF model is expected to have superior generalization performance compared with an RBF network trained using a two-stage procedure where in the first stage usupervised learning is applied. This superiority is also clearly illustrated in the experimental results discussed in later section.

2.4 Comparison between PRBF and Separate Mixtures

As stated previously, the training of the PRBF model follows different principles compared to unsupervised learning. The same holds when comparing PRBF with the separate mixtures

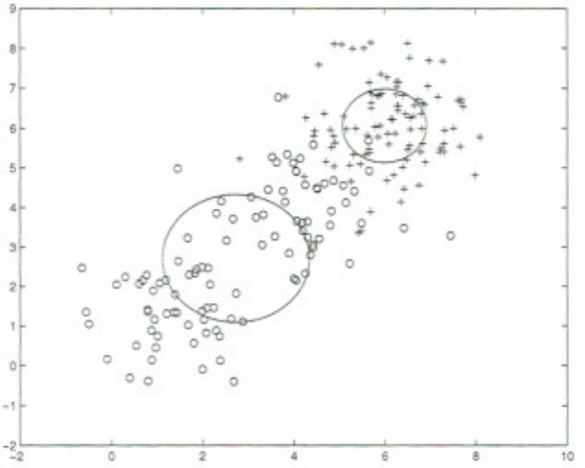


Figure 3: Illustrates the data of two classes and the location of the Gaussian kernels (represented by circles where the radius is equal to standard deviation) after (a) training a two-kernel PRBF with the EM algorithm and (b) training a two-kernel mixture model with EM.

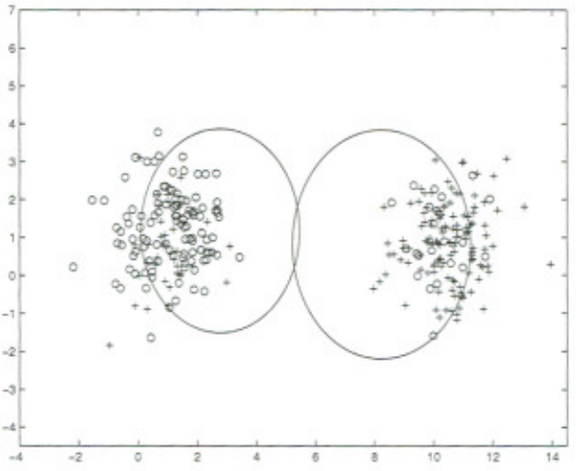
approach. There exist cases where PRBF provides results similar to separate mixtures. For example, such a case is the synthetic data set illustrated in Fig. 3. If we utilize a separate single kernel for estimating the conditional density of each class, we will obtain almost the same representation with that obtained from PRBF with two kernels. Nevertheless, in the following we discuss two cases where in the first one the PRBF represents the data more sufficiently than separate mixtures, while in the second the separate mixtures technique provides better representation of data than PRBF. We assume that both PRBF and separate mixtures utilize two kernels.

In the first example, assume that we have a two-class problem and the data set is displayed in Fig. 4. The data are arranged in two distinct regions, where in each region there exist many patterns of one class and few patterns of the opposite class. If we separately model each class conditional density by a single Gaussian kernel, then (as shown in Fig. 4(a)) we do not obtain a good representation of the actual densities. Obviously, this is due to the fact that a single kernel is not adequate to model the density of each class. On the other hand, the PRBF model with two kernels adjusts the kernel parameters so that the conditional densities are adequately modeled (Fig. 4(b)) and associates each kernel with both classes by appropriately adjusting the prior values. Note that in order to obtain the same representation using separate mixtures we need four kernels, that is two kernels for each mixture model. The above example implies that in cases where the data of different classes are highly overlapped, the PRBF may utilize the kernels more efficiently than the separate mixtures approach.

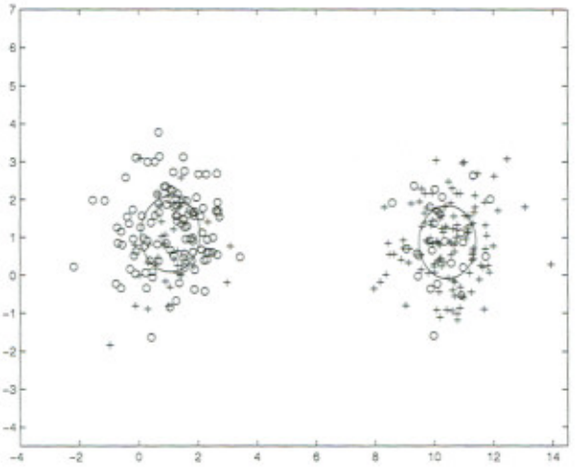
The data of the second example are displayed in Fig. 5 where we can observe that the first class data arise from one kernel, while the second class data arise from two distinct kernels. As shown in Fig. 5(a), the single kernel functions provided by a separate mixtures model represent the data more adequately compared to the PRBF solution. As illustrated in Fig. 5(b), PRBF does not manage to find a solution similar to that of Fig. 3 because the two regions of the second class are widely separated. This example shows that there exist cases where it is desirable to have a separate set of kernels devoted to represent data of each class. Finally, a general remark which can be drawn from the previous examples is that by combining properties of shared kernel models with those of separate mixtures, we can develop more general and efficient models for class conditional density estimation.

3 Intermediate Models between PRBF and Separate Mixtures

As pointed out in Section 2, the separate mixtures model can be considered as a constrained special case of the PRBF model. In the same way, the EM updates used for separate mixtures training can be obtained from the EM updates for training PRBF simply by setting some

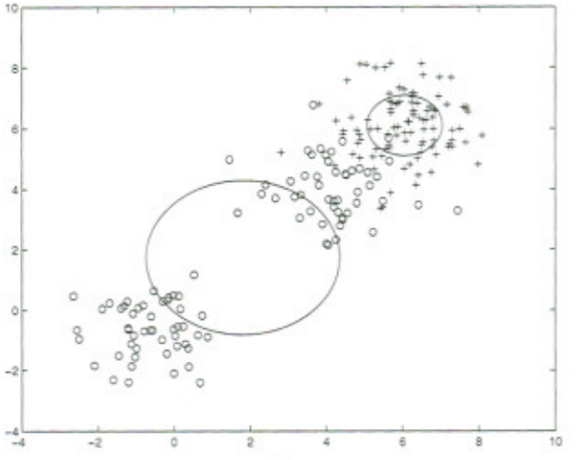


(a)

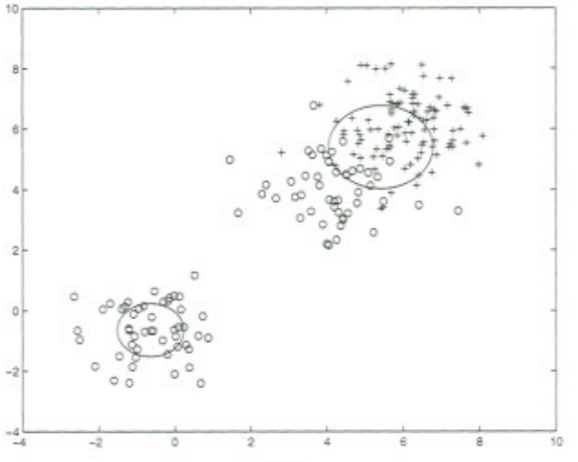


(b)

Figure 4: Displays the data for a two-class problem and the final solution found (a) using separate single kernels and (b) using a PRBF network with two kernels. Is is obvious that PRBF utilize the kernels so that to represent sufficiently the data.



(a)



(b)

Figure 5: Displays the data for a two-class problem and the final solution found (a) using two separate single kernels and (b) using a PRBF network with two kernels. In this case the single kernels give better representation of data than PRBF.

prior probabilities to zero.

We have also shown in the previous section that, depending on the data, the PRBF model may or may not provide better results compared to separate mixtures. From this point of view, it would be very interesting if we could express intermediate models between PRBF and separate mixtures for conditional density estimation. In this spirit, we have devised the λPRBF model described next.

The λPRBF model is actually a PRBF model, ie. they exhibit no difference in operation, once they have been trained. The main difference lies in the training process employed in the case of the λPRBF model.

In the λPRBF model there is an additional parameter λ (assuming values in $[0, 1]$), which is incorporated in the training process to control the degree of sharing of each kernel. More specifically, for a problem with K classes, the M kernels of a PRBF model are partitioned into K disjoint groups T_k, $k = 1, \ldots, K$, so that the group T_k corresponds to class C_k and $|T_1| + \ldots + |T_K| = M$. We wish that the kernels of group T_k would fully contribute to the density estimation of class C_k, while they would contribute less (depending on the value of λ) to the density estimation of the other classes. To express this preference we introduce the following function

$$p_\lambda(x|C_k) = \sum_{j \in T_k} \pi_{jk} p(x|j) + \lambda \sum_{j \notin T_k} \pi_{jk} p(x|j), \qquad k = 1, \ldots, K \tag{27}$$

where the expression $j \notin T_k$ denotes all kernels of the set $\bigcup_{k' \neq k} T_{k'}$. It is important to note that the priors π_{jk} satisfy the constraints (3), except for the case when λ is zero, where by definition it holds that

$$\sum_{j \in T_k} \pi_{jk} = 1, \qquad k = 1, \ldots, K. \tag{28}$$

Obviously, the function $p_\lambda(x|C_k)$ is not a probability density, (since $\sum_{j \in T_k} \pi_{jk} + \lambda \sum_{j \notin T_k} \pi_{jk} < 1$), except for the cases when λ is one or zero. This function is only defined for training purposes and must be distinguished from the class conditional density $p(x|C_k, \lambda)$ provided as output of the λPRBF model (after training). The function $p(x|C_k, \lambda)$ is computed in the usual PRBF way (2), ie. the parameter λ is not involved in the normal operation of the model. The parameter λ is included in the definition of $p(x|C_k, \lambda)$ just to denote its involvement is the training procedure.

The role of λ is to specify an a priori (user defined) preference that the model would be close to PRBF or to separate mixtures. Letting λ obtain values from one to zero, we move from the case of full sharing of kernels among classes (PRBF) to the case of no sharing of kernels (separate mixtures). More specifically, if λ is closer to zero, the kernels of group

T_k will be used more for representing the conditional density of the class C_k and less for representing the densities of the other classes. In the opposite case, when λ is closer to one, the kernels of T_k have more freedom to contribute to the estimation of all conditional densities. In other words, through the specification of λ, it is possible to impose a priori constraints to the grouped kernels, which express how much each group is available to contribute to the conditional density estimations of the other classes. In this sense, λ can be considered as a special type of hyperparameter, since it controls the adjustment of the rest of parameters.

Based on functions (27), we can introduce the posterior probability of a pattern x of class C_k having been generated from kernel j as follows

$$P_\lambda(j|C_k, x) = \begin{cases} \frac{\pi_{jk}p(x|j)}{p_\lambda(x|C_k)} = h_{jk}(x), & \text{if } j \in T_k \\ \frac{\lambda\pi_{jk}p(x|j)}{p_\lambda(x|C_k)} = \lambda h_{jk}(x), & \text{if } j \notin T_k \end{cases} \tag{29}$$

which satisfy

$$\sum_{j=1}^{M} P_\lambda(j|C_k, x) = \sum_{j\in T_k} h_{jk}(x) + \lambda \sum_{j\notin T_k} h_{jk}(x) = 1. \tag{30}$$

The introduced notation h_{jk} serves as a means of making the above definition and also the EM algorithm presented below more easily understandable. It is apparent that the posterior values are in general higher for the kernels of group T_k rather than for the rest of kernels since in the latter case the posteriors are not penalized by the parameter λ.

3.1 EM Algorithm for λPRBF

The training of the λPRBF model can be formulated as a maximization problem of the following function

$$\begin{aligned} L(\theta|\lambda) &= \log \prod_{k=1}^{K} \prod_{n=1}^{N_k} p_\lambda(x^n|C_k) \\ &= \sum_{k=1}^{K} \sum_{n=1}^{N_k} \log p_\lambda(x^n|C_k) \end{aligned} \tag{31}$$

subject to the constraints (3) concerning the priors. The above function can be regarded as a *penalized form* of the corresponding likelihood defined by (11). However, it must be noted that the penalties or parameter constraints are not expressed through the introduction of an additive term (a prior distribution) [5, 7], but are embedded in a novel way into the functional form of the original likelihood.

The same EM framework presented in Section 2.2 can also be applied in this case. The log-likelihood function of the complete data is

$$L_c(\theta|\lambda) = \sum_{k=1}^{K} \sum_{n=1}^{N_k} \left\{ \sum_{j\in T_k} z_j^n \log \pi_{jk} p(x^n|j) + \sum_{j\notin T_k} z_j^n \log \lambda\pi_{jk} p(x^n|j) \right\} \tag{32}$$

and the function Q to be maximized in the M-step is written as follows:

$$\begin{aligned} Q(\theta|\theta^{(t)},\lambda) &= \sum_{k=1}^{K}\sum_{n=1}^{N_k}\left\{\sum_{j\in T_k} P_\lambda^{(t)}(j|C_k,x^n)\log\pi_{jk}p(x^n|j) + \sum_{j\notin T_k} P_\lambda^{(t)}(j|C_k,x^n)\log\lambda\pi_{jk}p(x^n|j)\right\} \\ &= \sum_{k=1}^{K}\sum_{n=1}^{N_k}\left\{\sum_{j=1}^{M} P_\lambda^{(t)}(j|C_k,x^n)\log\pi_{jk}p(x^n|j) + \log\lambda\sum_{j\notin T_k} P_\lambda^{(t)}(j|C_k,x^n)\right\} \\ &= \sum_{k=1}^{K}\sum_{n=1}^{N_k}\sum_{j=1}^{M} P_\lambda^{(t)}(j|C_k,x^n)\log\pi_{jk}p(x^n|j) + \log\lambda\sum_{k=1}^{K}\sum_{n=1}^{N_k}\sum_{j\notin T_k} P_\lambda^{(t)}(j|C_k,x^n). \end{aligned} \tag{33}$$

The second term of (33) does not contain any adjustable parameter since λ is fixed parameter and therefore can be discarded. Using (29) the M-step requires the maximization of the function

$$Q(\theta|\theta^{(t)},\lambda) = \sum_{k=1}^{K}\sum_{n=1}^{N_k}\left\{\sum_{j\in T_k} h_{jk}^{(t)}\log\pi_{jk}p(x^n|j) + \lambda\sum_{j\notin T_k} h_{jk}^{(t)}\log\pi_{jk}p(x^n|j)\right\}. \tag{34}$$

Maximizing (34) is straightforward and it can be carried out in a similar way to that presented in Section 2.2. Finally, the following update equations are obtained:

$$\mu_j^{(t+1)} = \frac{\sum_{n=1}^{N_k} h_{jk}^{(t)}(x^n)x^n + \lambda\sum_{k'\neq k}\sum_{n=1}^{N_{k'}} h_{jk'}^{(t)}(x^n)x^n}{\sum_{n=1}^{N_k} h_{jk}^{(t)}(x^n) + \lambda\sum_{k'\neq k}\sum_{n=1}^{N_{k'}} h_{jk'}^{(t)}(x^n)} \tag{35}$$

$$\Sigma_j^{(t+1)} = \frac{\sum_{n=1}^{N_k} h_{jk}^{(t)}(x^n)w^n + \lambda\sum_{k'\neq k}\sum_{n=1}^{N_{k'}} h_{jk'}^{(t)}(x^n)w^n}{\sum_{n=1}^{N_k} h_{jk}^{(t)}(x^n) + \lambda\sum_{k'\neq k}\sum_{n=1}^{N_{k'}} h_{jk'}^{(t)}(x^n)} \tag{36}$$

$$\pi_{jk'}^{(t+1)} = \begin{cases} \frac{1}{N_{k'}}\sum_{n=1}^{N_{k'}} h_{jk'}^{(t)}(x^n), & \text{if } k'=k \\ \frac{\lambda}{N_{k'}}\sum_{n=1}^{N_{k'}} h_{jk'}^{(t)}(x^n), & \text{if } k'\neq k \end{cases} \tag{37}$$

where $j\in T_k$, $k=1,\ldots,K$ and w^n abbreviates the expression $(x^n-\mu_j^{(t+1)})(x^n-\mu_j^{(t+1)})^T$. The above equations actually differ from the corresponding of Section 2.2 only in the definition of the posterior probabilities which now are given by (29). An interesting issue is that the penalty mechanism (realized through λ) affects only the E-step of the algorithm. This differs in principle from the case of other penalized EM procedures where the penalties (expressed through a separate prior distribution) affect the M-step, while the calculation of the function Q remains unchanged [5].

Finally, the EM algorithm for training λPRBF is summarized as follows:

1. Specify the number of kernels M and the initial parameter values.

2. Set the parameter λ to a fixed value and specify the groups of kernels T_k, $k=1,\ldots,K$.

3. E-step: For each training point $(x^n, k^n) \in X$ compute the quantities $h^{(t)}_{jk^n}(x^n)$, $j = 1, \ldots, M$, from (29) using the current parameter values.

4. M-step: Find the new parameter vector $\theta^{(t+1)}$ from equations (35), (36) and (37) respectively.

5. Iterate going to step 2 until a local maximum of the log-likelihood (31) is reached.

It is straightforward to verify that for $\lambda = 0$ the above algorithm reduces to K independent EM procedures associated with the separate mixtures case, where the conditional density of the class C_k is modelled by a mixture containing the kernels of group T_k. Also in this case the special constraints concerning priors (28) are explicitly satisfied due to and (30) and (37). In the opposite extreme case where $\lambda = 1$, the update equations reduce to those corresponding to PRBF (Section 2.2).

3.2 Averaging over λ

From the previous presentation, it is obvious that the employment of the λPRBF model requires the specification of the parameter λ. Nevertheless, it is not clear how we can find a optimal value for this parameter. Probably, we could define a suitable prior distribution (or a regularization term) over the parameters and jointly maximize the likelihood and the prior to find an optimal λ value. However, the specification of such a prior function seems to be problem dependent according ot the discussion of Section 2.4. Therefore, we have implemented an alternative scheme that is based on a multi-net approach that combines the decisions of several models [11]. More specifically, we train several λPRBF networks for different λ values. To classify a new pattern we combine for each class (through averaging) the density estimations $p(x|C_k, \lambda)$ provided by the several models.

Performing averaging over λ is also motivated by a Bayesian perspective [9]. In the Bayesian fremwork, once the posterior distribution of the model parameters has been inferred, then any model related quantity can be computed through integration of this quantity with respect to the posterior distribution. In our case, once the distribution $p(\lambda|X)$ (for a given data set X) has been specified, then the conditional density $p(x|C_k)$ for a new pattern x^{N+1} can be exressed as

$$p(x^{N+1}|C_k) = \int_0^1 p(x^{N+1}|C_k, \lambda) p(\lambda|X) d\lambda. \tag{38}$$

Now, if we choose a set of values $\{\lambda_i,\ i = 1, \ldots, L\}$ for the parameter λ and obtain the corresponding estimations of the class conditional densities through training the λPRBF

model (for each value λ_i), then we can approximate (38) with the following average:

$$p(x^{N+1}|C_k) \approx \frac{1}{L}\sum_{i=1}^{L} p(x^{N+1}|C_k, \lambda_i). \tag{39}$$

In next section it is shown that performing averaging using few λ_i values leads in some cases to significant improvement of generalization performance.

4 Experimental Results

To assess the classification performance of the proposed shared kernel models, we have conducted a series of experiments on well-known classification data sets. We have implemented and tested the λPRBF network for various choices of the parameter λ. The form of kernel functions we used in all experiments is that of spherical Gaussians, (ie. $\Sigma_j = \sigma_j^2 I$) defined as

$$p(x|j) = \frac{1}{\sqrt{(2\pi\sigma_j^2)^d}} \exp\left\{-\frac{||x-\mu_j||^2}{2\sigma_j^2}\right\}. \tag{40}$$

Furthermore to illustrate the idea of integrating out the parameter λ, we also implemented the modular approach, where simple averaging is performed as described in equation (39). In addition, for typical comparison purposes, we have used the implementation of two-stage training for classical RBF networks available in the Netlab toolbox [8]. According to this implementation, 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 computed by a solving a linear system. However, it must be stressed that our purpose is mainly to test the λPRBF network as tool for class conditional Gaussian mixture modeling and not to perform comparisons with classification models that are based on function approximation (as is the RBF model).

In our experiments we have considered data sets the ESPRIT Basic Research Project ELENA (no. 6891) [4]. We have selected one artificial data set (Clouds) and two real data sets, namely Satimage and Phoneme.

To assess the performance of the several models for each problem we have selected the 5-fold cross-validation method. For each problem the original set was divided into five independent parts (holdouts), where each holdout was created using randomly selected patterns from the original set. Moreover, care was taken so that each part maintained the original proportions among the data of different classes (ie. the class priors). Using these holdouts, five pairs of training and test sets were constructed by keeping one of them for testing and joining the other four to form a training set. For each problem the results reported in the tables correspond to the average test error for the five pairs of training and test sets. We

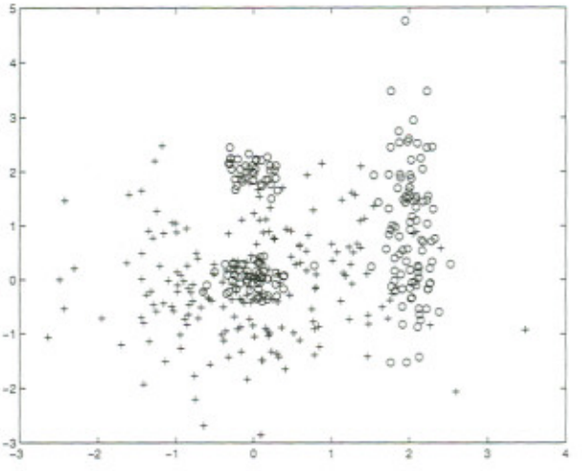


Figure 6: Illustrates the data of the clouds data set.

present results for several numbers of kernel functions which in all cases are multiples of the number of classes. We adopted this convention, because we would like the groups used by λPRBF to contain an equal number of kernels, since we assumed no prior information concerning the complexity of the data of each class. The kernels of group T_k were initialized using training patterns of the corresponding class C_k, and all models were tested under the same parameter initialization. Moreover, the bold numbers in each table indicate the model that provided best average performance for a specific number of kernels.

We first tested the algorithms using the two-dimensional *Clouds* data set (with two classes), which consists of 5000 patterns of two classes with equal class proportions. As illustrated in the graphical representation of this data set (Fig. 6) the two classes are highly overlapped. As Table 1 indicates, the RBF network provides high error, due to the improper way with which unsupervised learning for hidden layer places the kernels in the data space. On the other hand, the PRBF ($\lambda = 1$) gives the best generalization performance for almost all numbers of kernels.

The Satimage data set contains 6435 36-dimensional patterns belonging to six classes. The ELENA database provides also a five-dimensional description of this data set which was obtained using discriminant factorial analysis. This five-dimensional version of Satimage we use in our experiments. Performance results are displayed in Table 2. Table 3 displays the corresponding performance results for the Phoneme data set which contains 5404 five-dimensional patterns belonging to two classes.

From the presented experimental results, it is clear that the λPRBF network is more effective than the classical RBF network. Moreover, there is no clear conclusion that can be

	Number of kernels				
Algorithm	8	10	12	14	16
RBF	23.5	23.2	22.94	22.04	21.95
$\lambda = 0$	11.84	11.16	10.74	10.66	10.56
$\lambda = 0.25$	11.92	11.18	10.84	10.76	10.68
$\lambda = 0.5$	**11.26**	10.94	10.76	10.68	10.6
$\lambda = 0.75$	11.32	11.14	**10.66**	10.72	10.6
$\lambda = 1$	**11.26**	**10.72**	10.68	**10.52**	**10.54**
Averaging	11.48	10.9	10.72	10.66	10.64

Table 1: Results on Clouds data set.

	Number of kernels				
Algorithm	8	10	12	14	16
RBF	24.5	24.57	24.00	24.12	24.08
$\lambda = 0$	21.27	20.59	20.20	20.53	20.24
$\lambda = 0.25$	22.38	20.81	**19.85**	**19.94**	**20.03**
$\lambda = 0.5$	21.75	21.03	20.74	21	20.64
$\lambda = 0.75$	21.57	21.53	22.06	21.42	21.27
$\lambda = 1$	21.51	21.46	21.62	21.53	21.42
Averaging	**20.94**	**20.44**	20.33	20.64	20.35

Table 2: Results on Phoneme data set.

drawn concerning the performance of the PRBF ($\lambda = 1$) and the separate mixtures model ($\lambda = 0$). An important conclusion is that in many cases better performance results are obtained for intermediate values of λ and, also, that the multi-net approach, although more computationally expensive, constitutes a technique that on average seems to provide the best performance results.

5 Conclusions and Future Research

We have presented probabilistic models for class conditional density estimation, that are based on the idea of kernel sharing among the classes, which is in direct analogy with the classical RBF network. In this spirit we have presented the PRBF network and developed an EM algorithm for fast and effective PRBF training.

Moreover, we further extended the above idea and proposed a more general model (the λPRBF network) which allows for controlling the degree of sharing of grouped kernels among the classes. This general model constitutes a unifying framework for treating mixture models

	Number of kernels				
Algorithm	12	18	24	30	36
RBF	16.51	15.85	14.07	14.28	14.15
$\lambda = 0$	15.14	14.89	13.97	13.45	13.56
$\lambda = 0.25$	15.87	14.90	14	12.95	12.74
$\lambda = 0.5$	15.90	15.14	14.12	13.28	12.99
$\lambda = 0.75$	16.29	15.62	15.17	14.42	14.34
$\lambda = 1$	15.87	15.65	15.15	14.70	14.28
Averaging	**14.4**	**14.04**	**13.56**	**12.71**	**12.28**

Table 3: Results on Satimage data set.

for classification and encompasses as special cases both the PRBF network (for $\lambda = 1$) and the traditional separate mixtures approach (for $\lambda = 0$). We also developed an EM algorithm for efficient training of the λPRBF network. Since the performance of the model depends drastically on the value of λ (which is problem dependent and must be specified by the user), we also proposed a multi-net approach where several models are constructed for different values of λ and the network outputs are combined to classify a new pattern.

Current and future research is focused on two directions. The first is the development of a more flexible model that will allow for the separate specification of the degree λ_{jk} with which the kernel j is allowed to contribute to the conditional density estimation of class C_k. Besides, it is of significant importance to develop training algorithms that will automatically adjust the value of λ. The second research direction is related to the development of algorithms that dynamically adjust the number of kernels. Specifying the number of basis functions is an important open research issue in RBF training and mixture modeling, and our aim is to check the adaptation and applicability of the several techniques proposed so far in the framework of the PRBF network [14, 15].

References

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

[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] R. O. Duda and P. E. Hart, *Pattern classification and Scene Analysis*. Wiley, New York, 1973.

[4] Datasets and technical reports available via anonymous ftp from: ftp.dice.ucl.ac.be/pub/neural-nets/ELENA/databases.

[5] P. J. Green, *On use of the EM algorithm for penalized likelihood estimation*, Journal of the Royal Statistical Society B, 52, 443-452, 1990.

[6] G. J. McLachlan and K. Basford, *Mixture models: Inference and applications to clustering*. Wiley, 1988.

[7] G. J. McLachlan and T. Krishnan, *The EM algorithm and Extensions*. Marcel Dekker, 1997.

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

[9] R. M. Neal, *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics (no. 118), Springer, New York, 1996.

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

[11] A. Sharkey, *Combining Artificial Neural nets*, Springer, London, 1999.

[12] M. Titsias, A. Likas. *A Probabilistic RBF network for Classification*, Proc. of International Joint Conference on Neural Networks, Komo, Italy, July 2000.

[13] D. M. Titterington, A. F. Smith and U.E. Makov. *Statistical Analysis of Finite Mixture Distributions*. Wiley, 1985.

[14] 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.

[15] 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.