An Incremental Training Method for the Probabilistic RBF Network

Constantinos Constantinopoulos and Aristidis Likas, *Senior Member, IEEE*

***Abstract*—The probabilistic radial basis function (PRBF) network constitutes a probabilistic version of the RBF network for classification that extends the typical mixture model approach to classification by allowing the sharing of mixture components among all classes. The typical learning method of PRBF for a classification task employs the expectation–maximization (EM) algorithm and depends strongly on the initial parameter values. In this paper, we propose a technique for incremental training of the PRBF network for classification. The proposed algorithm starts with a single component and incrementally adds more components at appropriate positions in the data space. The addition of a new component is based on criteria for detecting a region in the data space that is crucial for the classification task. After the addition of all components, the algorithm splits every component of the network into subcomponents, each one corresponding to a different class. Experimental results using several well-known classification data sets indicate that the incremental method provides solutions of superior classification performance compared to the hierarchical PRBF training method. We also conducted comparative experiments with the support vector machines method and present the obtained results along with a qualitative comparison of the two approaches.**

***Index Terms*—Classification, decision boundary, mixture models, neural networks, probabilistic modeling, radial basis function networks.**

I. Introduction

ONE of the fundamental problems in machine learning is the supervised classification. That is the task of constructing a classifier for previously unseen patterns, given a training set of already classified patterns. Each pattern belongs to one class, and the number of possible classes is known. The statistical approach to the classification problem is to construct a model that estimates the class conditional densities $p(\mathbf{x}|k)$ of the data and the respective prior probabilities $P(k)$ for each class k. Then, using Bayes' theorem, the posterior probabilities $P(k|\mathbf{x})$ can be computed

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

In order to classify an unknown pattern $\mathbf{x}$, we select the class with the highest posterior probability $P(k|\mathbf{x})$ as suggested by the Bayes rule.

Manuscript received April 20, 2005; revised October 21, 2005. This paper was supported in part by the European Union under the framework of the Heraklitos Program of the Operational Program for Education and Initial Vocational Training of the Third Community Support Framework of the Hellenic Ministry of Education, by national sources, and by the European Social Fund.

The authors are with the Department of Computer Science, University of Ioannina, GR 45110 Ioannina, Greece (e-mail: ccostas@cs.uoi.gr; arly@cs.uoi.gr).

Digital Object Identifier 10.1109/TNN.2006.875982

In the conventional statistical approach, each class density $p(\mathbf{x}|k)$ is estimated using a separate mixture model. A mixture model [1] is a linear combination q of density functions, where $q(\mathbf{x}) = \sum_{j=1}^{J} w_j f_j(\mathbf{x})$. The mixing coefficients w_j are nonnegative and sum to unit, while the mixing components f_j are usually Gaussian densities. This *separate mixtures* approach estimates the density of each class independently from the others, considering only the data of the specific class. In the case where we assume one Gaussian component centered at each data point, then the probabilistic model of Specht is obtained [2]. This model assumes too many mixture components, requires no training, and has a performance that highly depends on the heuristic specification of the radius (σ) of the Gaussian components.

The probabilistic radial basis function (PRBF) network [3]–[5] constitutes an alternative approach for class conditional density estimation. It is an RBF-like neural network [6] adapted to provide output values corresponding to the class conditional densities $p(\mathbf{x}|k)$. Since the network is RBF, the components (hidden units) are shared among classes and each class conditional density is evaluated using not only the corresponding class data points (as in the case of separate mixtures) but also all the available data points. In order to train the PRBF network, the expectation–maximization (EM) algorithm for likelihood maximization can be applied [4], [7]–[9]. In addition, it has been shown [10] that the generalization performance is improved if after training the components are split in a certain way, so that new subcomponents are created that are not shared among classes. We refer to this approach as the *hierarchical PRBF* training method.

A significant issue in PRBF training is the initialization of the component parameters, since it affects the convergence point of the EM. EM is a local search algorithm, and thus is guaranteed to converge to a local maximum of the likelihood that possibly lies away from the global maximum; see [11] for more discussion. The influence of initialization on PRBF performance is also confirmed by our experimental results, provided in Section IV. A partial solution is to do multiple restarts of the EM with different initializations. Another approach is the application of the k-means algorithm (also employed in RBF network training) to obtain sensible initial parameter values. However, the problem is then transferred on how to initialize the k-means. Moreover, our motivation is to tackle the initialization problem in a way that benefits classification. This is not possible with any clustering algorithm, since it does not take into account class information.

Considering the problem of EM initialization and its influence on training performance, we propose an incremental training method for the hierarchical probabilistic RBF network, where components are sequentially added at appropriately selected positions in the data space. The main idea is the placement of the components near the decision boundary. For a

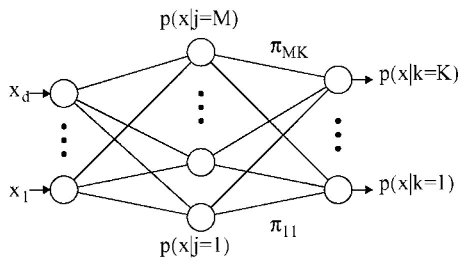

Fig. 1. Probabilistic RBF network.

given classifier, the decision boundary separates the data space in nonoverlapping regions that are assigned to different classes. We expect that a good estimation of the class densities around the decision boundary is sufficient to ensure good classification performance. Recent methods that focus on the decision boundary of the RBF classifier are described in [12] and [13]. The proposed method is deterministic, does not depend on the initialization of the network, and can be easily combined with model selection criteria in order to select the appropriate number of components. Experimental results on several well-known data sets illustrate that the approach is superior to the hierarchical PRBF training method (employing component splitting) [10] and demonstrates comparable performance to methods based on support vector machines (SVMs) [14].

It must be noted that several techniques [15]–[19] have been proposed on incremental training of the RBF network. They are also based on the idea of gradually adding neurons during training. However, these methods are mainly for sequential data (online learning) and for *regression* (or *function approximation*) problems. It is not straightforward to adapt these methods in the framework of the probabilistic RBF, which is a *statistical* approach to *classification* problems. It is fundamentally different both in terms of the model used (which is a Gaussian mixture model) and in terms of the learning task, which is classification and not regression. Consequently, the approach we have employed is quite different.

The following section summarizes the PRBF network and the splitting methodology. Section III describes the proposed incremental method, while in Section IV, comparative experimental results are presented using several data sets from the UCI repository [20]. Section V contains discussion and conclusions.

II. The Probabilistic RBF Network

A. Model Description and Learning

Consider a classification problem with K classes, where K is known and each pattern belongs to only one class. We are given a training set $X = \{(\mathbf{x}^{(n)}, y^{(n)}), n = 1, \ldots, N\}$, where $\mathbf{x}^{(n)}$ is a d-dimensional pattern and $y^{(n)}$ is a label $k \in \{1, \ldots, K\}$ indicating the class of pattern $\mathbf{x}^{(n)}$. The original set X can be 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 k, i.e., $N_k = |X_k|$.

Assume that we have M component functions (hidden units), which are probability density functions. In the PRBF network (Fig. 1), all component density functions $f_j(\mathbf{x}) = p(\mathbf{x}|j)$ are utilized for estimating the conditional densities of all classes by considering the components as a common pool [3], [4]. Thus, each class conditional density function $p(\mathbf{x}|k)$ is modeled as a mixture model of the form

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

where $f_j(\mathbf{x})$ denotes the component density j, while the mixing coefficient π_{jk} represents the prior probability that a pattern has been generated from the density function of component j, given that it belongs to class k. The priors take nonnegative values and satisfy the following constraint:

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

Once the outputs $p(\mathbf{x}|k)$ have been computed, the class of data point $\mathbf{x}$ is determined using the Bayes rule, i.e., $\mathbf{x}$ is assigned to the class with maximum posterior $P(k|\mathbf{x})$ computed by (1). The required priors are $P_k = N_k/N$, according to the maximum likelihood solution.

It is also useful to introduce the posterior probabilities expressing our posterior belief that component j generated a pattern $\mathbf{x}$ given its class k. This probability is obtained using the Bayes theorem

$$P(j|\mathbf{x}, k) = \frac{\pi_{jk} f_j(\mathbf{x})}{\sum_{i=1}^{M} \pi_{ik} f_i(\mathbf{x})}. \tag{4}$$

In the following, we assume Gaussian component densities of the general form:

$$f_j(\mathbf{x}) = \frac{|\Sigma_j|^{-1/2}}{(2\pi)^{d/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \mu_j)^T \Sigma_j^{-1} (\mathbf{x} - \mu_j)\right\} \tag{5}$$

where $\mu_j \in \Re^d$ represents the mean of component j, while Σ_j represents the corresponding $d \times d$ covariance matrix. The whole adjustable parameter vector of the model consists of the mixing coefficients π_{jk} and the component parameters (means μ_j and covariances Σ_j), and we denote it by Θ.

It is apparent that the PRBF model is a special case of the RBF network, where the outputs correspond to probability density functions and the second layer weights are constrained to represent the prior probabilities π_{jk}. Given an RBF classifier with Gaussian hidden units, its kth output is $g_k(\mathbf{x}) = \sum_j w_{jk} \exp\{-(1/2)(\mathbf{x} - \mu_j)^T(\mathbf{x} - \mu_j)/\sigma^2\}$. If w_{jk} are nonnegative and $\sum_j w_{jk} = 1$, then g_k is a density function. Actually it is the class conditional density $p(\mathbf{x}|k)$ that we estimate through PRBF. Furthermore, the separate mixtures model can be derived as a special case of PRBF, if we assign each component j to one class k and set $\pi_{j\ell} = 0$ for all classes $\ell \neq k$.

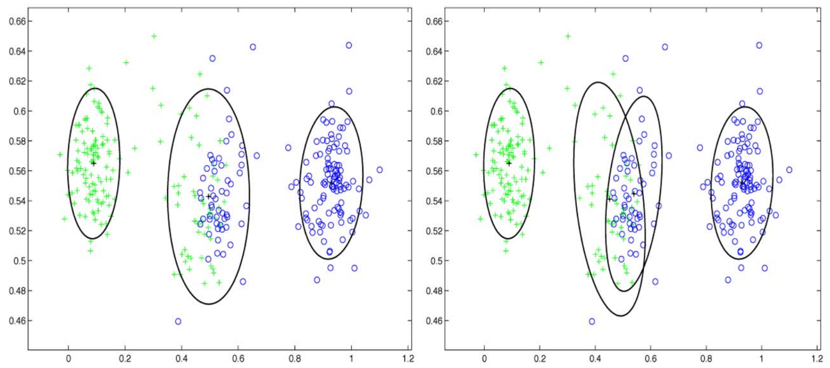

Fig. 2. Illustration of the component splitting process. The component in the middle is located in a region with data of two classes and is split into two subcomponents, each describing the data of one class. (Color version available online at http://ieeexplore.ieee.org.)

Regarding parameter estimation for the PRBF, the EM algorithm can be applied for maximization of the likelihood

$$\mathcal{L}(\Theta) = \sum_{k=1}^{K} \sum_{\mathbf{x}\in X_k} \log p(\mathbf{x}|k). \tag{6}$$

EM is an iterative procedure with two steps at each iteration. During the *expectation* step, posterior probabilities $\rho_{\mathbf{x},k}^{(j)} = P^{(t)}(j|\mathbf{x},k)$ are computed using the current estimates of $\pi_{jk}^{(t)}$, $\mu_j^{(t)}$ and $\Sigma_j^{(t)}$ according to

$$\rho_{\mathbf{x},k}^{(j)} = \frac{\pi_{jk}^{(t)} f_j\left(\mathbf{x};\mu_j^{(t)},\Sigma_j^{(t)}\right)}{\sum_{i=1}^{M} \pi_{ik}^{(t)} f_i\left(\mathbf{x};\mu_i^{(t)},\Sigma_i^{(t)}\right)}. \tag{7}$$

During the *maximization* step, the new estimates of the component parameters are updated according to

$$\mu_j^{(t+1)} = \frac{\sum_{k=1}^{K}\sum_{\mathbf{x}\in X_k} \rho_{\mathbf{x},k}^{(j)}\mathbf{x}}{\sum_{\ell=1}^{K}\sum_{\mathbf{x}\in X_\ell} \rho_{\mathbf{x},\ell}^{(j)}} \tag{8}$$

$$\Sigma_j^{(t+1)} = \frac{\sum_{k=1}^{K}\sum_{\mathbf{x}\in X_k} \rho_{\mathbf{x},k}^{(j)}\left(\mathbf{x}-\mu_j^{(t)}\right)\left(\mathbf{x}-\mu_j^{(t)}\right)^T}{\sum_{\ell=1}^{K}\sum_{\mathbf{x}\in X_\ell} \rho_{\mathbf{x},\ell}^{(j)}} \tag{9}$$

$$\pi_{jk}^{(t+1)} = \frac{1}{N_k}\sum_{\mathbf{x}\in X_k} \rho_{\mathbf{x},k}^{(j)}, \qquad k=1,\ldots,K. \tag{10}$$

The EM updates eventually will converge to a local maximum of the likelihood.

B. The Hierarchical Approach

In [10], a hierarchical mixture model method for classification has been proposed. This training method proceeds in two stages: in the first stage *(EM-stage),* a PRBF network with a fixed number of components M is trained using the EM updates (7)–(10). After the completion of this stage of training, there may be components of the network located to regions with overlapping among classes. This happens if we have underestimated the number M of components. In order to increase the generalization performance of the network, it is suggested in [10] to split each component. So in the second stage *(splitting stage)* of the training method, every PRBF component j is split into subcomponents jk corresponding to the classes of the problem. This is achieved by evaluating the posterior probability $P(j|\mathbf{x},k)$ [using (4)] for each component to define if it is responsible for patterns of more than one class. Therefore, we compute $P(j|\mathbf{x},k)$ for every pattern $\mathbf{x}\in X$ and check if $\sum_{\mathbf{x}\in X_k} P(j|\mathbf{x},k) > 0$ for more than one class k. If this happens, then we remove it from the network and add a separate component for each class. So finally every subcomponent describes only one class. Splitting a component j, the resulting subcomponent of class k is a Gaussian probability density function $f_{jk} = p(\mathbf{x}|j,k)$ with mean μ_{jk}, covariance matrix Σ_{jk}, and mixing weight π_{jk}. These parameters are estimated according to

$$\pi_{jk} = \frac{1}{N_k}\sum_{\mathbf{x}\in X_k} P(j|\mathbf{x},k) \tag{11}$$

$$\pi_{j\ell} = 0 \qquad \forall \ell \neq k \tag{12}$$

$$\mu_{jk} = \frac{\sum_{\mathbf{x}\in X_k} P(j|\mathbf{x},k)\mathbf{x}}{\sum_{\mathbf{x}\in X_k} P(j|\mathbf{x},k)} \tag{13}$$

$$\Sigma_{jk} = \frac{\sum_{\mathbf{x}\in X_k} P(j|\mathbf{x},k)(\mathbf{x}-\mu_{jk})(\mathbf{x}-\mu_{jk})^T}{\sum_{\mathbf{x}\in X_k} P(j|\mathbf{x},k)}. \tag{14}$$

After splitting, for each class k, there are M_k components with nonnegative mixing coefficients, and the class conditional density is

$$p(\mathbf{x}|k) = \sum_{j=1}^{M_k} \pi_{jk} f_{jk}(\mathbf{x}), \qquad k=1,\ldots,K. \tag{15}$$

Using the above equations, the components whose region of influence contains subregions with data of different classes are split into class-specific subcomponents. The parameters of each subcomponent are determined by the data of the respective class that belong to the region of the component (i.e., they have high posterior value). Fig. 2 provides a characteristic example illustrating the effect of the splitting operation. A remark that can be made is that, from a classification point of view, a full effect exploitation of the splitting operation is achieved if the components of the PRBF network have been placed into regions containing the decision boundary between classes (see Fig. 3). This remark led us to the development of an incremental method for

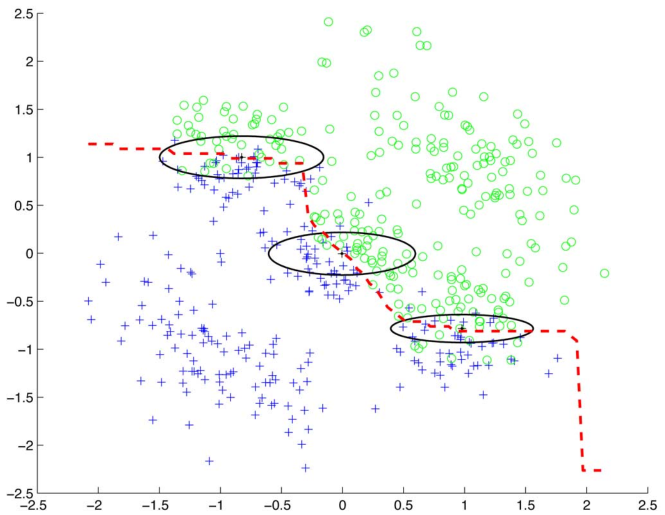

Fig. 3. Desirable placement of components on the decision boundary (shown with dashed line) in the first stage of hierarchical training. A subsequent splitting of each component will provide a satisfactory solution to the classification problem. (Color version available online at http://ieeexplore.ieee.org.)

placing the components of the PRBF network constructed in the first stage (EM training).

It has been shown in [10] that the addition of the splitting stage on the one hand guarantees the increase of the likelihood and on the other hand leads to considerable improvement in generalization performance compared to the PRBF network obtained in the first stage. However the solution of the EM-stage may be inferior, due to the local search performed by the EM algorithm. Since we are interested in the interpretation of the PRBF components as clusters of data, we would like to avoid cases where EM has converged to a solution where one component covers more than one cluster or one cluster is covered by more than one component. Standard EM can be trapped in such inferior solutions, and this depends solely on its initialization.

To deal with this problem, we propose an *incremental* method for the EM-stage that overcomes the initialization problem of EM. During the first stage, starting with one component, we incrementally add components to the network in a deterministic way. During the second stage we split all the components to obtain the final PRBF network.

III. The Incremental Training Method

The proposed incremental training method applies to the first stage of hierarchical training (EM-stage). As already noted, it is reasonable to place the components in regions containing patterns belonging to more than one class. This sensible placement is expected to lead, after the splitting stage, to a network with good classification performance.

Consider a PRBF network with M components during the first stage of training. In order to construct a network with $M+1$ components, the procedure of component addition involves global and local search in the parameter space to define the parameters of the new component. During global search, the algorithm searches among a set of candidate regions in the data space to place the new component and selects the most appropriate candidate according to certain criteria. Then, during local search, the EM algorithm is used to adjust the parameters of the resulting network with $M+1$ components. This procedure of sequential component addition starts with one component and is repeated until some stopping condition is met.

A. *Component Addition*

In this section, we present the procedure we have developed to specify the parameters of the new component. Assuming a network with M components and parameter vector Θ_M, the conditional density of each class k is $p(\mathbf{x}|k;\Theta_M)$. In the case where a new component $j = M+1$ is added with density $f_{M+1}(\mathbf{x})$, each new class conditional density $p(\mathbf{x}|k;\Theta_{M+1})$ is defined as a mixture of the current model $p(\mathbf{x}|k;\Theta_M)$ and the new component $f_{M+1}(\mathbf{x})$

$$p(\mathbf{x}|k;\Theta_{M+1}) = (1-\alpha_k)p(\mathbf{x}|k;\Theta_M) + \alpha_k f_{M+1}(\mathbf{x}) \quad (16)$$

where α_k $(k = 1,\ldots,K)$ are the mixing weights for the new component and $\alpha_k \in (0,1)$. This is analogous with the incremental training procedure called Greedy-EM proposed in [21] for unsupervised probability density estimation. Using the above combination formula, the resulting network is again a PRBF network. The log-likelihood $\mathcal{L}(\Theta_{M+1})$ of the model with $M+1$ components is

$$\mathcal{L}(\Theta_{M+1}) = \sum_{k=1}^{K}\sum_{\mathbf{x}\in X_k}\log\{(1-\alpha_k)p(\mathbf{x}|k;\Theta_M)+\alpha_k f_{M+1}(\mathbf{x})\}. \quad (17)$$

Let PRBF(M) denote the PRBF model after M components have been added and PRBFsplit(M) denote the resulting model after splitting the components of PRBF(M). The incremental training algorithm can be outlined as follows.

```
1) Set $M := 1$. Compute the one-component model
   PRBF(1) as follows:
```

$$\mu_1 = \frac{1}{|X|}\sum_{k=1}^{K}\sum_{\mathbf{x}\in X_k}\mathbf{x} \quad (18)$$

$$\Sigma_1 = \frac{1}{|X|}\sum_{k=1}^{K}\sum_{\mathbf{x}\in X_k}(\mathbf{x}-\mu_1)(\mathbf{x}-\mu_1)^T \quad (19)$$

$$\pi_{1k} = 1, \quad k = 1,\ldots,K. \quad (20)$$

```
2) Find the parameters of the new component
   density $f_{M+1}(\mathbf{x})$ and the corresponding
   mixing weights $\alpha_k$, considering $p(\mathbf{x}|k;\Theta_M)$
   fixed. In the case where a new component
   cannot be added terminate the incremental
   procedure and go to step 7).
3) Initialize the model with $M+1$ components
   using (16).
4) Apply the EM update equations to the model
   with $M+1$ components until convergence to
   obtain the PRBF($M+1$) model.
5) Set $M := M+1$.
6) If $M \leq M_{max}$, go to step 2).
7) Compute the PRBFsplit($M$) model, using
   (11)to (15).
```

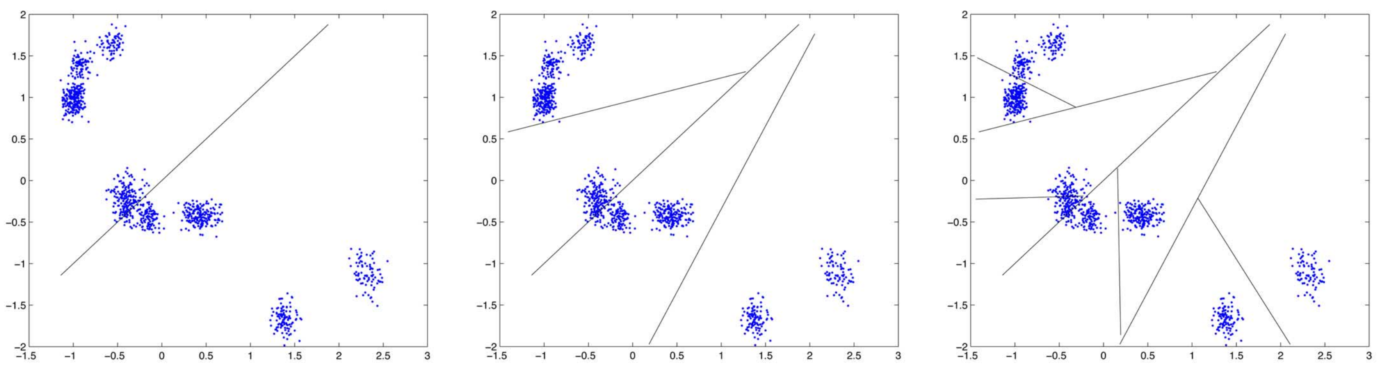

Fig. 4. Successive partitioning of an artificial dataset overlapping partitions using the kd-tree method. All 14 partitions illustrated in the three graphs are considered to specify candidate parameter vectors. (Color version available online at http://ieeexplore.ieee.org.)

Steps 1)–6) constitute the PRBF construction phase, while Step 7) corresponds to the splitting phase where the PRBFsplit model is determined. It is also apparent that the incremental procedure terminates either in Step 2) (in the case where it not possible to identify an appropriate region in the data space to place the new component) or in Step 6) when a prespecified maximum number of PRBF components has been added.

B. Where to Place the New Component?

The crucial task during component addition takes place at Step 2), where the parameters of the new component are determined through a search procedure among a set of candidate solutions. We can summarize this procedure in three steps.

```
a) Define a set of candidate components using
   a data partitioning technique.
b) Adjust the parameters of the candidate
   components.
c) Use a selection criterion to choose the
   candidate component that will be added.
```

Since it is not possible to directly specify a single good component to add, we define a set of *candidate initial* component parameters and further adjust the parameters using *partial EM*. The best parameter values (μ, Σ, a_k) obtained according to a specific criterion are considered as the final component parameters used in Step 3) of the method.

Let M be the current number of PRBF components. In order to determine the candidate initial component parameters, we partition the data set X into M subsets $S_j = \{\mathbf{x}|P(j|\mathbf{x}) > P(i|\mathbf{x}), \forall i \neq j\}$, one for each component based on the posterior probabilities $P(j|\mathbf{x})$. The posteriors are computed marginalizing class labels

$$P(j|\mathbf{x}) = \sum_{k=1}^{K} P(j|\mathbf{x}, k)P(k) \tag{21}$$

with $P(k) = N_k/N$ being the prior probability of class k. For each of the M sets S_j, a subset of candidate components is created by partitioning its data using the *kd-tree* approach. A *kd-tree* [22] defines a recursive partitioning of the data space into disjoint subspaces. It is a binary tree, where the data associated with any nonterminal node are partitioned using a *cutting hyperplane* to specify the successors nodes. To partition the data points of a node, we have followed the approach used in [23]: the cutting hyperplane is defined to be perpendicular to the direction of the principal component of the data corresponding to the node. Fig. 4 illustrates the partitioning stages for an artificial data set. The procedure of recursive partitioning is applied until level four (tree depth), and we consider all tree nodes (not only leaf nodes) to define overlapping subsets of S_j (i.e., 14 subsets for each component j). The statistics (sample mean and covariance) of each of the $14M$ subsets constitute candidate initial parameter values for the component $M+1$. The values of α_k are set equal to $\pi_{jk}/2$ for the subsets derived from partitioning the subset S_j.

In order to further adjust the mean, the covariance, and the mixing weights of each candidate component, we apply *partial EM* updates, where only the parameters of the new component are updated. This constrained local optimization is fast (usually a couple of iterations are sufficient), and the resulting component lies near the decision boundary. Let Θ_M denote the parameter vector of PRBF(M) that is considered fixed during partial EM. In the *expectation* step of partial EM, we compute the posterior probabilities $r_{\mathbf{x},k} = P^{(t)}(j = M+1|\mathbf{x}, k)$ using the current estimates of $\alpha_k^{(t)}$ and $\theta^{(t)} = \{\mu_{M+1}^{(t)}, \Sigma_{M+1}^{(t)}\}$, according to

$$r_{\mathbf{x},k} = \frac{\alpha_k^{(t)} f_{M+1}(\mathbf{x}; \theta^{(t)})}{\left(1 - \alpha_k^{(t)}\right) p(\mathbf{x}|k; \Theta_M) + \alpha_k^{(t)} f_{M+1}(\mathbf{x}; \theta^{(t)})}. \tag{22}$$

During the *maximization* step, the component parameters are updated as

$$\mu_{M+1}^{(t+1)} = \frac{\sum_{k=1}^{K} \sum_{\mathbf{x}\in X_k} r_{\mathbf{x},k}\, \mathbf{x}}{\sum_{\ell=1}^{K} \sum_{\mathbf{x}\in X_\ell} r_{\mathbf{x},\ell}} \tag{23}$$

$$\Sigma_{M+1}^{(t+1)} = \frac{\sum_{k=1}^{K} \sum_{\mathbf{x}\in X_k} r_{\mathbf{x},k}\, (\mathbf{x}-\mu_{M+1}^{(t)})(\mathbf{x}-\mu_{M+1}^{(t)})^T}{\sum_{\ell=1}^{K} \sum_{\mathbf{x}\in X_\ell} r_{\mathbf{x},\ell}} \tag{24}$$

$$\alpha_k^{(t+1)} = \frac{1}{|X_k|} \sum_{\mathbf{x}\in X_k} r_{\mathbf{x},k} \qquad k = 1, \ldots, K. \tag{25}$$

After applying partial EM steps for each candidate component, we obtain the final set of candidate initial parameter vectors $\theta^l = \{\mu^l, \Sigma^l, a_k^l\}$ for component $M+1$. One of these vectors will be selected using the following procedure.

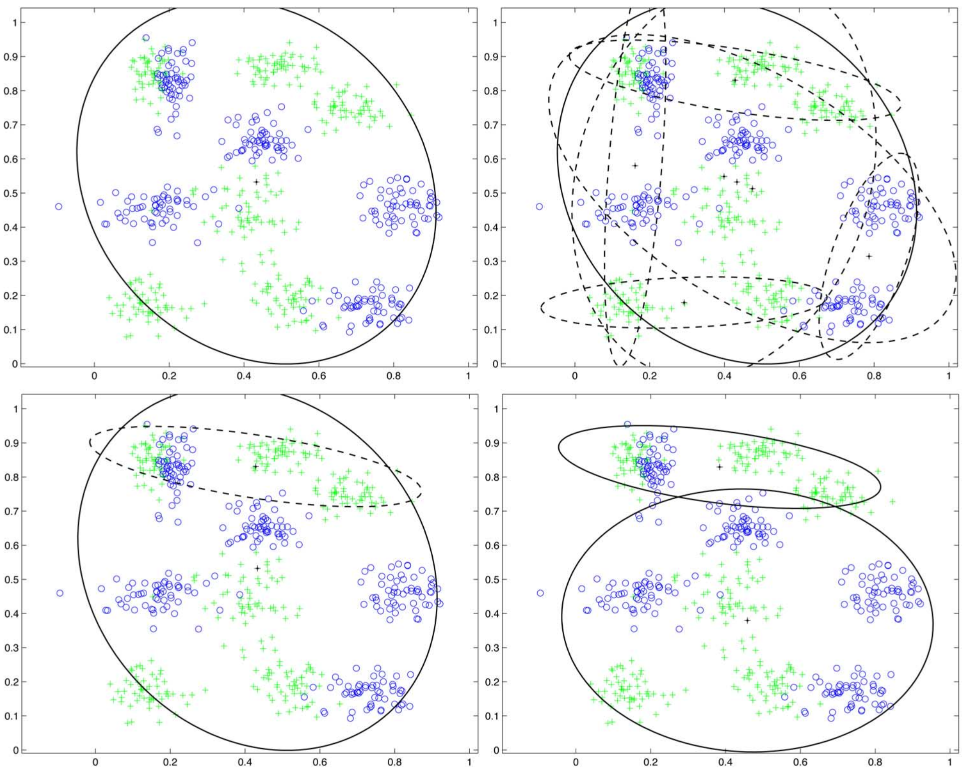

Fig. 5. Illustration of the incremental procedure for adding the first two PRBF components. The components of the network are drawn with solid lines and the candidate components are presented using dotted lines. (Color version available online at http://ieeexplore.ieee.org.)

We have already mentioned that we wish the new component to be placed at regions in the data space containing examples of more than one class. A way to quantify the degree to which a candidate component satisfies this property is to compute *the change of the log-likelihood for class* k caused by the addition of the candidate new component l with density $f_l = p(\mathbf{x};\theta^l)$ according to (16). So we define the change $\Delta\mathcal{L}_k^l$ for class k as

$$\begin{aligned}\Delta\mathcal{L}_k^l &= \frac{1}{N_k}\left(\mathcal{L}_k\left(\Theta_{M+1}^l\right) - \mathcal{L}_k(\Theta_M)\right)\\ &= \frac{1}{N_k}\sum_{\mathbf{x}\in X_k}\log\left\{1-\alpha_k+\alpha_k\frac{p(\mathbf{x};\theta^l)}{p(\mathbf{x}|k;\Theta_M)}\right\} \quad (26)\end{aligned}$$

where $\Theta_{M+1}^l = \Theta_M \cup \theta^l$. Based on the values $\Delta\mathcal{L}_k^l$, we search among the candidate components l to determine those whose addition causes an *increase in the log-likelihood for at least two classes*. Such candidates lie in a region containing data of more than one class, consequently on the decision boundary. In order to find the best candidate, we retain the components that increase the log-likelihood of at least two classes and discard the rest. For each retained component l, we add the positive $\Delta\mathcal{L}_k^l$ terms to compute the *total increase of the log-likelihood* $\Delta\mathcal{L}_l$. The candidate $l^\star$ whose value $\Delta\mathcal{L}_{l^\star}$ is maximum is added to the current model PRBF(M) if this maximum value is higher than a prespecified threshold (set equal to 0.01 in all experiments). Otherwise, we consider that the attempt to add a new component is unsuccessful and terminate the first stage of training with a PRBF model with M components. This threshold value refers to the increase in likelihood after a new component is added. Eventually, after the addition of many components, the increase tends to zero and the addition stops. We set this threshold empirically, to decide when the increase of likelihood is negligible and avoid redundant component additions. After experimentation with several data sets we concluded that changes of the likelihood smaller than this scale do not affect classification performance and also do not lead to premature termination of network growing.

Fig. 5 graphically illustrates the procedure of component addition. The top left graph is the model with one component [Step 1)]. The top right graph displays the single component and six of the 14 candidate components after [Step 2b)]. The best candidate selected in [Step 2c)] is depicted in the bottom left graph. Finally the bottom right graph is the model with two components PRBF(2), after application of the EM [Step 4)].

It must be noted that, due to the EM update equations [Step 4)], it is possible for an added component to move away from the decision boundary where it has initially been placed. This may happen in the case where there exists a region in the data space containing data of a single class that is not adequately covered by any of the existing components. However, this does not constitute a problem for our method since the next component to be added is very likely to be placed and remain at the initial position of the previous component.

Finally, a very desirable feature of the proposed incremental method is that for training a network with M components (PRBF(M)), the method builds all the intermediate models PRBF(m) with $m = 1,\ldots,M$ components. Since for each $m = 1,\ldots,M$, the PRBFsplit(m) model is constructed directly from PRBF(m), any model selection procedure to search for the best PRBFsplit(m) model can be implemented very

TABLE I
CHARACTERISTICS OF THE DATA SETS

	BLD	*PID*	*Iris*	*Veh*	*Glass*	*Wave*	*Wine*	*Thyr*
patterns	345	768	150	846	214	5000	178	215
features	6	8	4	18	9	21	13	5
classes	2	2	3	4	6	3	3	3

efficiently, compared to the conventional PRBF method where a separate EM run is needed for every value of $m = 1, \ldots, M$ to construct PRBF(m) (in fact many runs for every m are needed due to the EM initialization problem). An analogous drawback also holds with the SVM method where several runs with different values of SVM "hyperparameters" (e.g., width of the RBF kernels) are necessary to determine the "hyperparameter" values that lead to the SVM model with the best performance.

IV. Experimental Results

The proposed incremental hierarchical training method for the PRBF network (denoted as *incremental PRBFsplit*) has been compared with the standard hierarchical PRBF method [10] (denoted as *PRBFsplit*) and with SVM. We used the OSU SVM Classifier Matlab Toolbox version 3.0 [24]. The core part of this toolbox is based on LIBSVM version 2.33 [25]. We considered several benchmark data sets from the UCI repository [20], namely, Bupa Liver Disorder (BLD), Pima Indian Diabetes (PID), Iris, Vehicles (Veh), Glass, Waveform (Wave), Wine, and Thyroid (Thyr). The number of patterns, the number of features, and the number of classes for each dataset are summarized in Table I.

For each data set, in order to obtain an estimate of the generalization error, ten-fold cross-validation was used, i.e., ten experiments were conducted, with one of the folds used for testing and the remaining nine folds for training. In each experiment, nine networks were constructed for different values of M and covariance type (discussed later), using one of the nine folds as a validation set and the remaining eight folds for the incremental training process (training set). The combination of M and covariance type that provided the best average classification performance on the nine runs was selected, and subsequently evaluated using the data of the test fold. Exactly the same validation procedure and folds were also used for adjusting the hyperparameters in the SVM approach and for model selection in standard PRBFsplit method. We employed an SVM model with RBF kernels $K(\mathbf{x}, \mathbf{y}) = \exp\{-\gamma|\mathbf{x}-\mathbf{y}|^2\}$. For multiclass problems, the 1 *versus* 1 scheme [26] was used to reduce them to binary-class problems. The hyperparameters to be adjusted were the inverse width γ of the kernel and the cost C of constraint violation. For each experiment, in order to determine the value of (C, γ), we initialized $C = 1$ and searched through the set $\{2^s | s = -50, \ldots, 50\}$ to find the value of γ that minimizes classification error on the validation set. Fixing γ to this optimal value, we searched again for the best value of C in the set $\{2^s | s = -50, \ldots, 50\}$. Obviously this method does not search for all possible combinations of (C, γ) but it was selected in order to maintain reasonable execution times. We have also experimented with the exhaustive search method, over a grid in the space of (C, γ). The classification performance was almost the same, while the execution time was 10–70 times higher.

TABLE II
MEAN ERROR (%) AND THE STANDARD DEVIATION (IN PARENTHESES) OF THE THREE METHODS USING TEN-FOLD CROSS-VALIDATION

	PRBFsplit	*Incremental PRBFsplit*	*SVM*
BLD	31.3 (7.8)	28.4 (5.2)	30.5 (5.3)
PID	25.3 (4.6)	24.1 (5.8)	22.7 (4.6)
Iris	3.7 (5.2)	2.0 (3.2)	3.3 (4.7)
Veh	20.4 (3.6)	14.3 (5.17)	16.5 (5.5)
Glass	34.4 (10.3)	28.6 (8.2)	29.1 (9.5)
Wave	14.4 (1.5)	14.2 (1.9)	13.3 (1.2)
Wine	2.2 (2.8)	0.5 (1.75)	1.1 (3.5)
Thyr	7.0 (4.5)	4.6 (6.1)	5.5 (5.2)

An important issue affecting the performance of the PRBF network concerns the specification of the covariance type. We used three parametrizations of the covariance matrix: a symmetric positive definite matrix (called $full$), a positive diagonal matrix (called $diagonal$), and a positive diagonal matrix with all diagonal elements equal (called $spherical$). The type of covariance affects the number of parameters of the model and the shape of the components; thus we have to compromise. A full covariance allows arbitrary component shapes but may increase unnecessarily model complexity and lead to overfitting. Moreover, for small data sets with a lot of features, the use of full parametrization could lead to singular covariances. To deal with this issue, for every fold we have performed three runs of the proposed method, each of them with the corresponding type of covariance. In this way the sequences of models PRBFsplit$^{\text{full}}(m)$, PRBFsplit$^{\text{diag}}(m)$, and PRBFsplit$^{\text{spherical}}(m)$ were constructed, and we selected the model giving the best performance in the validation set. It is apparent that for different folds, the selected model could differ in the number of components and the covariance type. Concerning the standard PRBFsplit method, for each value of $m = 1, \ldots, M_{\max}$ and covariance type, five runs of EM were conducted with different initializations. The model with the maximum likelihood value was selected as the network PRBF(m) that was subsequently split. Regarding the maximum number of components $M_{\max}$ it was set equal to 30 in all experiments. In most cases the proposed training algorithm terminated earlier due to the inability to identify additional significant components to be added to the model.

Table II provides the obtained mean value and standard deviation of the generalization error for the three methods, that is, the average percent of misclassified patterns for all the cross-validation test sets. It must be noted that for each data set, exactly the same division in folds was used for training with the three methods. It is clear that the incremental method is superior to the standard PRBFsplit method. With respect to SVM, the performance results are comparable. In some data sets the proposed method provides better results, while in some others the SVM appears to be slightly superior. It does not seem to be possible to draw reliable conclusions regarding the superiority of one method over the other in terms of generalization error.

TABLE III
AVERAGE EXECUTION TIME (IN SECONDS) AND NUMBER OF COMPONENTS/VECTORS FOR INCREMENTAL PRBFsplit AND SVM USING TEN-FOLD CROSS-VALIDATION

	Incremental PRBFsplit		*SVM*	
	time	*components*	*time*	*vectors*
BLD	33.8	4.5	39.7	224.0
PID	61.4	4.3	164.1	413.2
Iris	16.2	3.0	19.0	50.7
Veh	87.5	4.0	617.0	438.2
Glass	60.6	16.9	20.1	161.4
Wave	118.5	4.6	11357.1	2381.8
Wine	28.1	3.6	30.2	88.4
Thyr	45.3	3.8	19.1	69.4

Table III provides comparative experimental results for our method and SVM. The results concern execution speed and the number of allocated components or support vectors. In terms of the number of components, our method uses a significantly smaller number compared to the SVM support vectors. As discussed in the next section, this is due to the fact that our method is region-based, i.e., the model is described by a relatively small number of wide regions (specified by the regions of influence of the Gaussian components). In terms of execution speed, the comparison highly depends on how many SVM training runs are conducted to search for appropriate values of the SVM hyperparameters (C, γ). The SVM execution times refer to the case where local search is performed to adjust these SVM hyperparameters, and the results are comparable to our method. As already noted, in the case where exhaustive search is performed for the SVM hyperparameters, the execution times are 10–70 times higher. Execution times refers to Matlab implementations of both our method and the SVM method (in the latter case the Ohio State University SVM Classifier toolbox has been used) on the same PC platform. A qualitative comparison of the two approaches is presented in the next section.

V. DISCUSSION AND CONCLUSION

We have proposed an incremental training method for the probabilistic RBF network that overcomes the initialization problem of the standard EM algorithm and provides networks with superior generalization performance. The proposed approach is based on the sensible placement of new network components in regions that are of interest from the classification point of view, i.e., regions containing data of several classes (decision boundaries). These components are then split into class-specific subcomponents providing a refined estimation of class densities in the regions of interest. Experimental results indicate that the method constitutes a competitive and much faster alternative to the SVM approach that deserves to be examined when building a classification system.

The proposed incremental hierarchical training method exhibits several interesting differences compared to SVM. First, it is a statistical approach respecting the *generative* paradigm as opposed to the *discriminative* paradigm supported by SVM and typical feedforward neural models (MLP and RBF). Moreover, the following differences can be mentioned.

- It is an inherently multiclass method, while typical SVMs are basically two-class classification methods and multiclass extensions require significant further elaboration.
- The PRBF network performs classifications based on class conditional probabilities $P(k|\mathbf{x})$; therefore it is straightforward to compute uncertainty about the decisions made or to exploit these probabilities in a probabilistic decision-making framework.
- The PRBF method can naturally provide class ranking based on the corresponding class conditional probabilities.
- The PRBF approach relies its classification decisions on an estimation of the density of each class in the region of the input pattern; therefore the decisions made are intuitively more easily interpreted. On the other hand, the SVM method draws decision boundaries in the kernel space; therefore its classification decisions are more difficult to be interpreted.
- The PRBF model is described by a relatively small number of regions (specified by the regions of influence of the Gaussian components), i.e., it is a region-based method. On the other hand, the SVM model is described by a much higher number of points (support vectors), i.e., it is a point-based method. This explains the fact that the number of PRBF components is significantly smaller compared to support vectors as shown in Table III.
- In the incremental PRBF method, it is much easier and faster to perform model selection, since all models are constructed in a single run.
- Since it is a density-based approach, the PRBF model is expected to exhibit worse performance in problems with very small number of examples and very high number of features, as happens for examples in many bioinformatics classification problems (e.g., classification of gene expression data).
- The SVM approach does not exhibit numerical difficulties (e.g., singular covariance matrices) that are sometimes encountered in training with Gaussian components.

We believe that there is room for further investigation mainly on the issue of determining the set of candidate components to be added at each step. Also, we are going to examine the use of probabilistic principal component analyzers [27] instead of Gaussian mixture components.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] G. McLachlan and D. Peel, *Finite Mixture Models*. New York: Wiley, 2000.
[2] D. F. Specht, "Probabilistic neural networks," *Neural Netw.*, vol. 3, pp. 109–118, 1990.
[3] M. K. Titsias and A. Likas, "A probabilistic RBF network for classification," in *Proc. IEEE Int. Joint Conf. Neural Networks*, 2000, vol. 4, pp. 238–243.
[4] ——, "Shared kernel models for class conditional density estimation," *IEEE Trans. Neural Netw.*, vol. 12, no. 5, pp. 987–997, Sep. 2001.
[5] ——, "Class conditional density estimation using mixtures with constrained component sharing," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 7, pp. 924–928, Jul. 2003.

[6] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.

[7] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood estimation from incomplete data via the EM algorithm," *J. Roy. Statist. Soc. B*, vol. 39, no. 1, pp. 1–38, 1977.

[8] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*. New York: Wiley, 1997.

[9] T. Mitchell, *Mach. Learn.*. New York: McGraw-Hill, 1997.

[10] M. K. Titsias and A. Likas, "Mixture of experts classification using a hierarchical mixture model," *Neural Comput.*, vol. 14, no. 9, pp. 2221–2244, 2002.

[11] C. Biernacki, G. Celeux, and G. Govaert, "Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models," *Comput. Statist. Data Anal.*, vol. 41, no. 3–4, pp. 561–575, 2003.

[12] Z. K. Mao and G. Huang, "Neuron selection for RBF neural network classifier based on data structure preserving criterion," *IEEE Trans. Neural Netw.*, vol. 16, no. 6, pp. 1531–1540, Nov. 2005.

[13] Y. Oyang, S. Hwang, Y. Ou, C. Chen, and Z. Chen, "Data classification with radial basis function networks based on a novel kernel density estimation algorithm," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 225–236, Jan. 2005.

[14] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[15] J. Platt, "A resource-allocating network for function interpolation," *Neural Comput.*, vol. 3, no. 2, pp. 213–225, 1991.

[16] V. Kadirkamanathan and M. Niranjan, "A function estimation approach to sequential learning with neural networks," *Neural Comput.*, vol. 5, no. 6, pp. 954–975, 1993.

[17] L. Yingwei, N. Sundararajan, and P. Saratchandran, "A sequential learning scheme for function approximation using minimal radial basis function neural networks," *Neural Comput.*, vol. 9, no. 2, pp. 461–478, 1997.

[18] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 57–67, Jan. 2005.

[19] ——, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks," *IEEE Trans. Syst., Man, Cybern. B*, vol. 34, 6, pp. 2284–2292, Dec. 2004.

[20] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz, UCI repository of machine learning databases [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html 1998

[21] N. A. Vlassis and A. Likas, "A greedy-EM algorithm for Gaussian mixture learning," *Neural Process. Lett.*, vol. 15, pp. 77–87, 2002.

[22] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[23] A. Likas, N. A. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognit.*, vol. 36, pp. 451–461, 2003.

[24] J. Ma, Y. Zhao, and S. Ahalt, Ohio State Univ. SVM Classifier Matlab Toolbox (Ver. 3.00) [Online]. Available: http://www.ece.osu.edu/~maj/osu_svm

[25] C. Chang and C. Lin, LIBSVM—A library for support vector machines [Online]. Available: http://www.csie.ntu.edu.tw/ cjlin/libsvm

[26] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," *J. Mach. Learn. Res.*, vol. 1, pp. 113–141, 2000.

[27] M. Tipping and C. Bishop, "Mixtures of probabilistic principal component analysers," *Neural Comput.*, vol. 11, no. 2, pp. 443–482, 1999.

Constantinos Constantinopoulos received the B.Sc. and M.Sc. degrees from the Computer Science Department, University of Ioannina, Greece, in 2000 and 2002, respectively, where he is currently working towards the Ph.D. degree.

His research interests include neural networks, machine learning, and Bayesian reasoning.

Aristidis Likas (S'91–M'96–SM'03) received the diploma degree in electrical engineering and the Ph.D. degree in electrical and computer engineering from the National Technical University of Athens, Greece, in 1990 and 1994, respectively.

Since 1996, he has been with the Department of Computer Science, University of Ioannina, Greece, where he is currently an Associate Professor. His research interests include neural networks, machine learning, statistical signal processing, and bioinformatics.