[5] K. Tomiyasu, "Preliminary results of a spectral analysis of simulated complex pulse response history of a synthetic aperture radar pixel," *IEEE Trans. Geosci. Remote Sensing*, vol. GE-22, pp. 577–581, Nov. 1984.
[6] F. T. Ulaby and J. J. Van Zyl, "Wave properties and polarization," in *Radar Polarimetry for Geoscience Applications*, F. T. Ulaby and C. Elachi, Eds. Norwood, MA: Artech House, 1990, ch. 1.

A Technique for the Selection of Kernel-Function Parameters in RBF Neural Networks for Classification of Remote-Sensing Images

Lorenzo Bruzzone and Diego Fernández Prieto

***Abstract*—In this paper, a supervised technique for training radial basis function (RBF) neural network classifiers is proposed. Such a technique, unlike traditional ones, considers the class memberships of training samples to select the centers and widths of the kernel functions associated with the hidden neurons of an RBF network. The result is twofold: a significant reduction in the overall classification error made by the classifier and a more stable behavior of the classification error versus variations in both the number of hidden units and the initial parameters of the training process.**

***Index Terms*—Image analysis, neural networks, pattern analysis, remote sensing.**

I. Introduction

Since the end of the 1980's, when the results of first experiences in the use of neural networks for classification of multisource remote-sensing data were published [1]–[3], several papers have appeared that stress the capability of neural networks for analyzing remotely sensed data. In particular, different models of neural networks have been proposed, among which, the multilayer perceptron (MLP) trained with the error backpropagation (EBP) learning algorithm is the most widely used [4]. The popularity of the MLP stems from the ability of this model to generate arbitrary decision boundaries in the feature space, provided that the network architecture relies on two or more hidden layers [5]. However, MLP's exhibit serious drawbacks and limitations (e.g., the slow convergence of the EBP learning algorithm, the potential convergence to a local minimum, the common chaotic behavior of nonlinear systems, and the inability to detect that a pattern to be classified has fallen into a region of the input space without training data) [6]. RBF neural networks [7]–[11] overcome some of these problems by relying on a rapid training phase, avoiding a chaotic behavior, and presenting systematic low responses to input patterns that have fallen into regions of the input space where there are no training samples. Such characteristics and the intrinsic simplicity of these networks render RBF neural classifiers an interesting alternative to classifiers based on other neural models.

Manuscript received May 29, 1998; revised August 10, 1998. This research was supported in part by the Italian Space Agency (ASI).

The authors are with the Department of Biophysical and Electronic Engineering, University of Genoa, I-16145, Genova, Italy (e-mail: lore@dibe.unige.it).

Publisher Item Identifier S 0196-2892(99)01970-1.

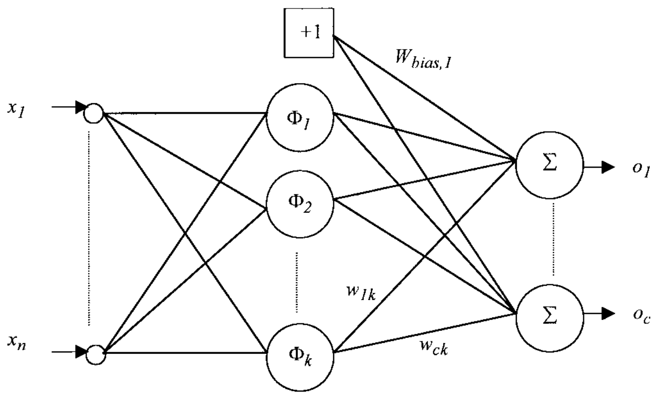


Fig. 1. Typical architecture of a classifier based on RBF neural networks.

However, the classification error made by RBF neural classifiers strongly depends on the selection of the centers and widths of the kernel functions associated with the hidden neurons of the network.

In this paper, we propose a simple, yet effective, training technique that, unlike traditional ones, selects the centers and widths of the kernel functions on the basis of the class-membership information available in the training set. Experimental results obtained on a multisource remote-sensing data set are reported that show the effectiveness of the proposed technique.

The paper is organized into six sections. Section II provides a brief description of RBF neural classifiers. Section III addresses the main problems associated with the use of traditional techniques for training the hidden layer of RBF neural classifiers. In Section IV, the proposed technique is detailed. The data set used for experiments is described in Section V, where experimental results are also reported. Finally, conclusions are drawn in Section VI.

II. RBF Neural Network Classifiers

Let us consider a classification problem in which the ith sample, described by an n-dimensional feature vector $\underline{x}_i = (x_1, \cdots, x_n)$, is to be assigned to one of c different land-cover classes $\Omega = \{\omega_1, \omega_2, \cdots, \omega_c\}$. In this context, the RBF network architecture for classification purposes (Fig. 1) consists of three layers: one input layer, one hidden layer, and one output layer. The input layer relies on as many neurons as input features. Input neurons just propagate input features to the next layer. Each neuron in the hidden layer is associated with a kernel function $\Phi_j(\cdot)$ (in this paper we shall consider a Gaussian function), characterized by a center $\underline{\mu}_j$ and a width σ_j. The output layer is composed of as many neurons as classes to be recognized. Each output neuron o_l computes a simple weighted summation over the responses of the hidden neurons for a given input pattern $\underline{x}_i$:

$$o_l(\underline{x}_i) = \sum_{j=1}^{k} w_{lj}\Phi_j(\underline{x}_i) + w_{\text{bias},\,l} \tag{1}$$

where k is the number of hidden neurons, w_{lj} represents the weight associated with the connection between the kernel function $\Phi_j(\cdot)$ and the output neuron o_l, and $w_{\text{bias},\,l}$ is the bias of the output neuron o_l.

The training of RBF neural classifiers is usually carried out in two steps: 1) the hidden layer is trained by selecting the centers and widths of the kernel functions associated with the hidden units; 2)

Tthe weights corresponding to the connections between the hidden units and the output units are fixed.

1) Several techniques have been proposed for the selection of the centers of the kernel functions [10]. Such a selection is typically performed by applying a clustering technique, like the *k-means* clustering algorithm [12], to the whole training set, without considering the class-membership information available.

   Once the centers of the kernel functions have been fixed, the selection of the widths of such functions is carried out. The width values control the generalization capabilities of the network [9]. The width of a given kernel function can be chosen as the standard deviation computed over all training samples included in the cluster associated with the kernel function considered. However, this strategy may lead to the generation of narrow not overlapping kernel functions that affect the generalization ability of the network. A widely used alternative lies in the use of the *p-nearest-neighbor* (*p*-nn) technique [9], which generates a certain overlapping of the kernel functions, thus improving the generalization capabilities of the classifier.

2) In the final stage of the training process, the weights corresponding to the connections between the hidden nodes and the output units are fixed. This is typically done by minimizing a *sum-of-squares* error function [10]. As a *sum-of-squares* error function is a quadratic function of the weights, its minimum can be found in terms of the solution of a set of linear equations given by a pseudo-inverse matrix [10]. This represents one of the major advantages of RBF neural networks as it avoids the need for a nonlinear optimization of the weights, which would require a high computational load.

In this training process, an efficient selection of the centers and widths of the kernel functions associated with the hidden neurons of the network is still an open issue in literature as this choice may strongly affect the errors made by the classifier. This problem is addressed in the following section.

III. On the Selection of the Kernel-Function Parameters

Kernel functions can be considered as processing elements that carry out a nonlinear transformation of the n-dimensional input space so as to increase the range of possible decision boundaries that separate different classes in the kernel-function space. Consequently, an RBF neural classifier can be represented as a simpler network with as many input neurons as kernel functions and as many output neurons as land-cover classes. As the output neurons of the network are characterized by a linear discriminant function (i.e., a simple weighted summation over the responses of the kernel functions), they generate linear decision boundaries (i.e., hyperplans) in the kernel-function space. Therefore, the performances of RBF neural classifiers strongly depend on the linear separability of classes in the k-dimensional space generated by the nonlinear transformations carried out by the k hidden units (i.e., kernel functions) of the network.

Two main factors affect the separability of classes in the kernel-function space: the presence of *mixed* kernel functions and the overlapping of kernel functions associated with different classes. In the following, both problems are faced.

A. Mixed Kernel Functions

Traditional techniques for selecting the centers of the kernel functions make use of clustering techniques applied to the whole training set, without considering the class-membership information about each training sample. Such a clustering process may lead to the generation of clusters composed of data points belonging to different classes (*mixed* clusters). This type of clusters gives rise to kernel functions (*mixed* kernel functions) that contribute to reducing the separability of classes in the kernel-function space. Fig. 2 shows a simple example in which training patterns belonging to the classes ω_1 and ω_2 are assigned to three different clusters: S_1, S_2, and S_m. As we can see, the kernel function $\Phi_m(\cdot)$ associated with the *mixed* cluster S_m leads to a nonseparable situation in which patterns belonging to different classes may be represented by the same vectors in the kernel function space. This can be explained if we consider that the center $\underline{\mu}_m$ of the *mixed* cluster S_m represents a prototype for the training samples located in the surroundings of $\underline{\mu}_m$ [10], irrespective of their class-memberships. As a consequence, the corresponding *mixed* kernel $\Phi_m(\cdot)$ may generate identical responses to input patterns belonging to different classes located in the surroundings of $\underline{\mu}_m$. Therefore, the kernel function $\Phi_m(\cdot)$ may reduce the separability of classes in the kernel function space, thus affecting the overall classification error made by the neural network classifier.

A further problem associated with the use of classical techniques for selecting the centers of the kernel functions of RBF classifiers is the unstable behavior of the classification error made by the classifier, depending on both the number of hidden units and the initial selection of the cluster centers. One of the causes of this oscillatory behavior is the random number of *mixed* clusters that are generated when we vary either the number of hidden units or the initial cluster centers. Small variations in these parameters may lead to different solutions of the clustering process (i.e., different numbers of *mixed* clusters), thus causing large variations in the classification errors made by RBF neural classifiers. These two problems make it very critical to select the initial architecture of the network and to initialize the training process.

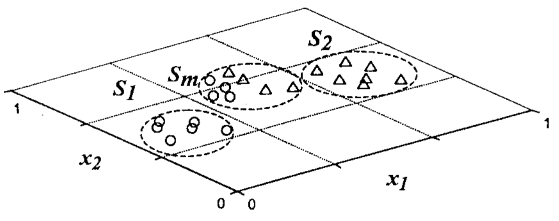
B. Overlapping of the Kernel Functions

As stated in Section II, the *p*-nn technique is the most widely used for the selection of kernel-function widths. Such a technique, which allows some overlapping of kernel functions, aims at increasing the generalization capabilities of the network. However, the overlapping of the kernel functions associated with different classes reduces the separability of such classes in the regions of the kernel-function space corresponding to the overlapping areas; as a consequence, the classification errors on samples located in these areas increase. This points out that the *p*-nn is not a suitable criterion for the selection of the widths of kernel functions located in boundary regions between classes.

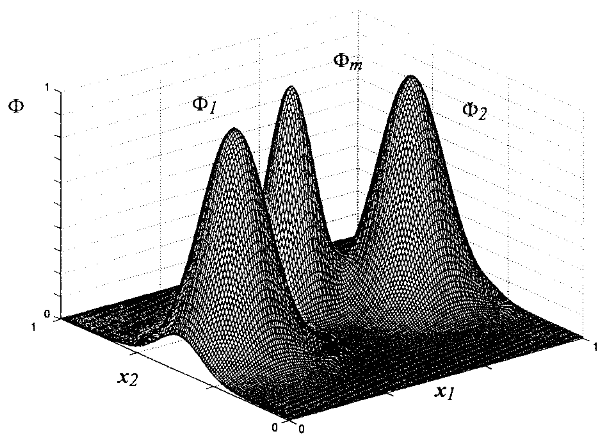
IV. Description of the Proposed Technique

The basic idea of the proposed technique is to carry out the training process of the hidden layer of RBF neural classifiers by taking into account the class-memberships of the training samples. In particular, clusters are generated by grouping training samples belonging to the same class in order to avoid the creation of *mixed* clusters. Moreover, the widths of the kernel functions are selected by using a supervised procedure aimed at limiting the widths of kernels located in boundary regions between classes while maintaining, at the same time, a certain overlapping inside the internal regions of each class. In the following, a detailed description of the proposed technique is provided.

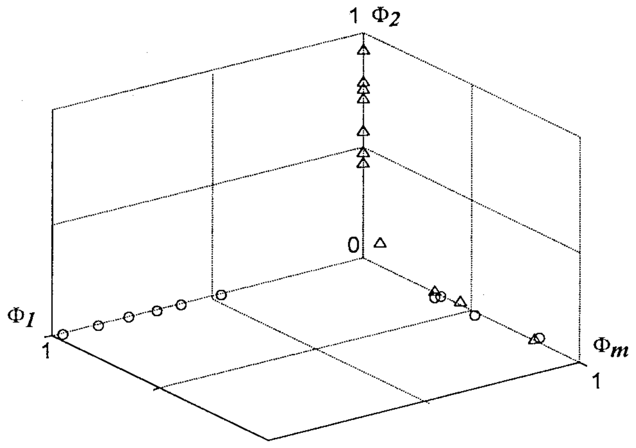
Let us assume the availability of a training set $T = \{\tau_1 \cup \tau_2 \cup \cdots \cup \tau_c\}$, where τ_i represents a subset of T containing all training samples $\underline{x}_j{}^i\,(j = 1, \cdots, N_i)$ belonging to the class ω_i. The proposed technique is subdivided into two sequential phases:



(a)



(b)



(c)

Fig. 2. Simple example, in which some training samples [in a two-dimensional (2-D) space] belonging to the classes ω_1 and ω_2 (o = class ω_1, Δ = class ω_2) are assigned to three different clusters S_1, S_2 and S_m. (a) Representation of the three clusters in the input space (S_m is a *mixed* cluster). (b) Kernel functions generated by the considered clusters. (c) Representation of the training samples in the kernel function space. The *mixed* kernel Φ_m (i.e., the kernel function generated by S_m) leads to nonlinearly separable situations.

computation of the kernel-function centers and computation of the kernel-function widths.

A. Computation of the Kernel-Function Centers

This phase aims at partitioning the N_i training samples belonging to τ_i into K_i disjoint subsets (i.e., clusters) $S_q^i\,(q = 1, \cdots, K_i)$ so that at the end of the process $\tau_i = \{S_1^i \cup S_2^i \cup \cdots \cup S_{K_i}^i\}\ \forall\, \tau_i \subset T$. Such a task is carried out by the following algorithm.

Step 1) Let $i = 1$.

Step 2) For the class ω_i, the number of centers K_i is chosen according to both the number of training samples N_i and the dimension of the input space n.

Step 3) The *k-means* clustering algorithm is applied to the subset τ_i. The centers $\underline{\mu}_q^i$ of the clusters $S_q^i \subset \tau_i\,(q = 1, \cdots, K_i)$ are initialized with different randomly chosen training samples belonging to τ_i. Then each training sample belonging to τ_i is assigned to the cluster nearest to it so that the sum of the Euclidean quadratic distances between the training points assigned to each cluster and the related cluster center is minimized. This means that the algorithm finds a local minimum of

$$E_i = \sum_{q=1}^{K_i} \sum_{\underline{x}_j{}^i \in S_q^i} \left\| \underline{x}_j{}^i - \underline{\mu}_q^i \right\|^2. \tag{2}$$

Then, the center $\underline{\mu}_q^i$ of each cluster S_q^i is updated by computing the barycenter of the cluster. The procedure is iterated until convergence is reached [12]. At the end of the clustering process, all training samples included in τ_i turn out to be assigned to different clusters so that $\tau_i = \{S_1^i \cup S_2^i \cup \cdots \cup S_{K_i}^i\}$. At this point, each cluster center $\underline{\mu}_q^i$ is associated with a kernel function $\Phi_q^i(\cdot)$.

Step 4) Update $i = i+1$. If $i \le c$ go to Step 1). Otherwise, END.

The above-described procedure generates c subsets of kernel functions that can be associated with specific land cover classes. This results in the elimination of the *mixed* clusters and hence in a better separability of classes in the kernel function space. In addition, the elimination of *mixed* kernel functions results in a more stable classification error versus variations in both the number of hidden neurons of the network and the initialization of the cluster centers.

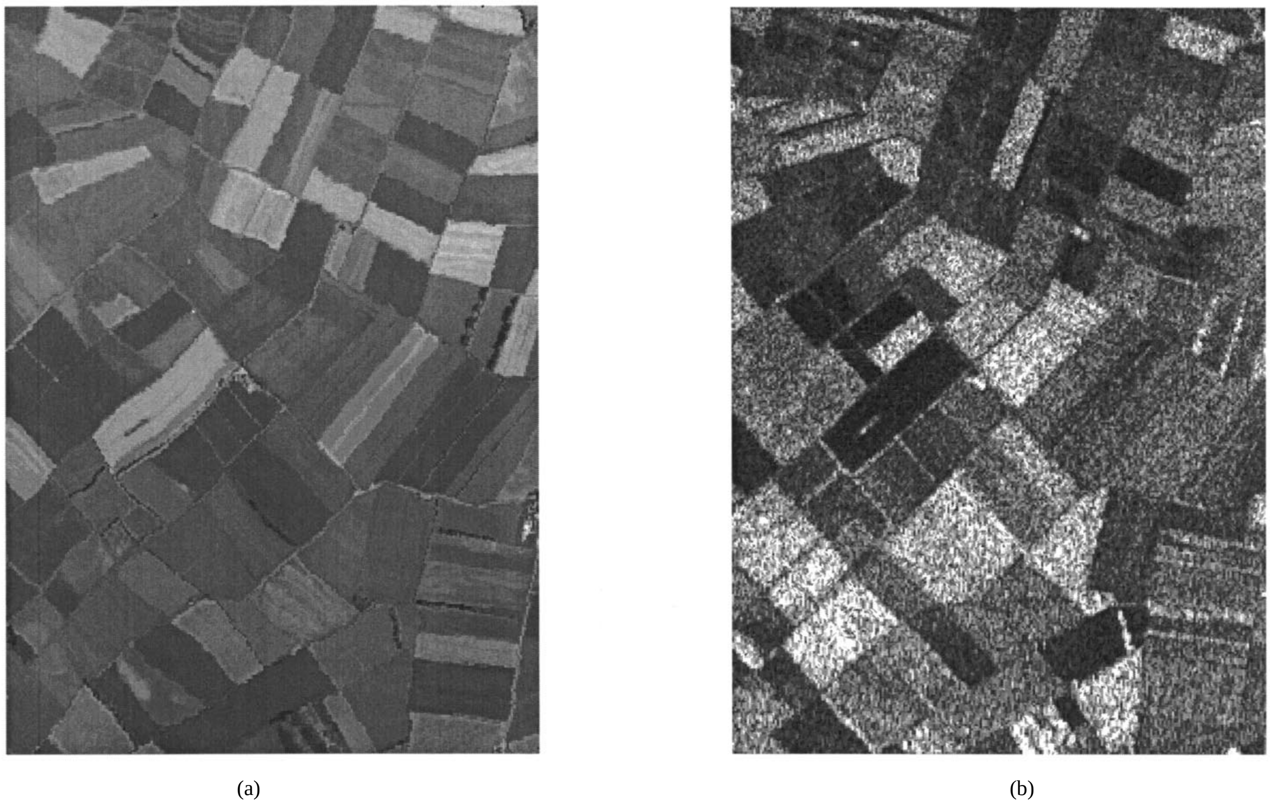


Fig. 3. Multisensor image utilized for the experiments: (a) channel 9 of the ATM sensor; (b) channel L-HV (L-band, HV-polarization) of the SAR sensor.

It is worth noting that, even though the proposed technique has been described in terms of the *k-means* clustering algorithm, the basic idea of the proposed approach could also be applied in the contexts of other clustering procedures (e.g., the *Isodata* clustering algorithm [12]).

B. Computation of the Kernel-Function Widths

The widths of the kernel functions are computed in accordance to a hybrid criterion. The width σ_q^i of the kernel function $\Phi_q^i(\cdot)$ (related to the class ω_i) is computed with the *p*-nn technique only if $\Phi_q^i(\cdot)$ is surrounded by M kernel functions associated with the class ω_i (i.e., if the M cluster centers closest to the center $\underline{\mu}_q^i$ belong to the class ω_i). Otherwise, the width σ_q^i is set to the standard deviation computed over all training samples belonging to the cluster S_q^i. This criterion aims at increasing generalization in the internal regions of each class (thanks to a certain overlapping of the kernel functions related to the same class). At the same time, it limits errors due to the overlapping of the kernel functions (associated with different classes) located in the boundary regions between classes (thanks to the narrower kernel functions resulting from choosing the widths as the standard deviation).

V. Experimental Results

A. Data Set Description

To carry out an experimental analysis to validate the proposed technique, we considered a multisource data set composed of images acquired in the same area by two different types of airborne sensors: a Daedalus 1268 airborne thematic mapper (ATM) scanner and a PLC-band, fully polarimetric, NASA/JPL SAR sensor. The selected data set refers to a section (250 $\times$ 350 pixels) of a scene acquired in an agricultural area near Feltwell, U.K. [13]. Fig. 3 shows channel 9 of the ATM sensor and channel L-HV (L-band, HV-polarization) of the SAR sensor. Images were registered by using the SAR image as a reference. The available ground truth was used to prepare a reference map to assess the classification error. We considered five land cover classes corresponding to five types of crops. The agricultural fields were randomly subdivided into two disjoint sets: 5124 training pixels were selected from the fields of one set and 5820 test pixels were taken from the fields of the other set. Fifteen channels were selected to form a feature vector for each pixel: we selected the six ATM channels corresponding to TM channels in the visible and infrared spectrum (except the thermal band) and the nine SAR channels in the PLC-band and HH-, HV-, VV-polarizations. The noise affecting the intensity values of the images was reduced by applying a simple running mean filtering to both the ATM (5 $\times$ 5 window) and the SAR (9 $\times$ 9 window) images.

B. Results and Discussion

To evaluate the effectiveness of the proposed training technique, three different experiments were carried out. In these experiments, the proposed technique was compared with the classical one in terms of classification accuracy, stability, and processing time. As suggested in the literature [9], the classical approach was implemented by using the *k-means* clustering algorithm (based on the Euclidean distance) and the *p*-nn criterion (with p equal to two). Concerning the proposed algorithm, parameters p and M were taken equal to two and three, respectively.

The first experiment compared the overall classification errors (versus the number of hidden units) made by the RBF classifier when using the proposed and the classical training techniques. Several trials were carried out, increasing the number of hidden neurons (and hence the number of kernel functions) from 15 to 100 (by steps

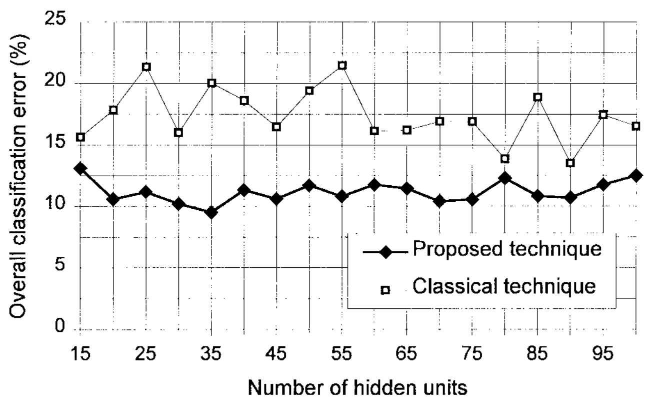


Fig. 4. Classification errors made on the test set by the RBF neural classifier trained by using both the classical and the proposed techniques versus the number of hidden units.

equal to five). For the sake of simplicity, an equal number of kernel functions for each class was chosen in all of the trials carried out with the proposed technique. Results, shown in Fig. 4, confirm that the presented technique significantly reduces the classification error made by the RBF neural classifier. In particular, the lowest classification error made by using the classical technique was equal to 13.5% (with 90 hidden units), whereas the minimum classification error made with the proposed technique decreased to 9.5% (with 35 hidden units). In addition, Fig. 4 points out that the classification error made with the classical technique follows an oscillatory behavior with regard to the number of hidden neurons considered. On the contrary, the proposed technique results in a more stable trend of the classification error, thus providing a better framework for choosing the architecture of the network.

The second experiment showed the stability of the overall classification error made by the RBF neural classifier trained by using both the proposed and the classical techniques, despite the random initialization of the cluster centers. Fifteen trials were carried out changing, in a random manner, the initial cluster centers for a given number of kernel functions. For this experiment, the number of kernel functions was fixed at 50 (i.e., ten per class when using the proposed technique). Table I gives the class by class means and standard deviations of the classification errors computed for the 15 trials carried out using both the proposed and the classical techniques. From this table, it is easy to deduce that the proposed technique, besides reducing the overall classification error, also shows a more stable behavior versus the random initialization of the cluster centers. In particular, the overall classification error made by the RBF classifier trained with the classical training technique was found to vary from 13.5 to 20.2%, the mean and the standard deviation being equal to 16.4% and 2.19, respectively. The overall classification error made by the classifier trained by using the proposed technique was found to range from 9.7 to 12.0%, the mean and the standard deviation being equal to 10.8% and 0.81, respectively. Table I also shows that the proposed technique results in a more balanced classification error on the classes, thanks to the better class descriptions provided by the kernel functions.

The third experiment compared the overall classification error of the RBF classifier trained by using both the proposed and the classical techniques with the errors made by an MLP, a probabilistic neural network (PNN) and a k-nearest neighbor (k-nn) nonparametric classifiers [13]. The phase of the architecture design and the trials carried out using the above-mentioned classifiers on the same data set, as considered in this paper, are described in [13]. Table II shows

TABLE I
STATISTICS (MEAN AND STANDARD DEVIATION) COMPUTED FOR THE CLASSIFICATION ERRORS MADE IN 15 TRIALS CARRIED OUT ON THE TEST SET BY USING THE RBF NEURAL CLASSIFIER TRAINED WITH BOTH THE PROPOSED AND THE CLASSICAL TECHNIQUES. TO CARRY OUT THE TRIALS, THE NUMBER OF HIDDEN UNITS WAS SET TO 50

Land-cover classes	Classification error			
	Classical training technique		Proposed training technique	
	Mean	St. deviation	Mean	St. deviation
Sugar beets	1.8%	1.97	0.7%	1.43
Stubble	19.6%	3.78	13.0%	4.29
Bare soil	42.5%	16.91	17.1%	1.01
Potatoes	26.0%	6.53	23.7%	0.84
Carrots	20.6%	6.19	13.6%	0.25
Overall	16.4%	2.19	10.8%	0.81

TABLE II
THE SMALLEST ERRORS ON THE CLASSIFICATIONS OF TEST PIXELS PERFORMED BY THE RBF NEURAL CLASSIFIER TRAINED WITH THE PROPOSED TECHNIQUE (P.T.) AND THE CLASSICAL TECHNIQUE (C.T.), AND BY THE MLP CLASSIFIER, THE PNN CLASSIFIER, AND THE k-nn CLASSIFIER

Land-cover classes	Number of pixels	RBF (P.T.)	RBF (C.T.)	MLP	PNN	k-nn
Sugar beets	2043	0.4%	2.0%	2.0%	2.2%	2.6%
Stubble	1371	13.5%	12.5%	15.4%	17.6%	11.6%
Bare soil	555	15.4%	17.8%	19.1%	20.4%	24.0%
Potatoes	884	15.4%	32.8%	19.1%	18.2%	13.6%
Carrots	967	13.8%	18.9%	11.8%	10.7%	12.9%
Overall	5820	9.5%	13.5%	10.4%	11.4%	10.2%

the best results obtained by the different classifiers in terms of the classification error on the test set. As we can see, even though the classification error made by the RBF classifier trained by using the proposed technique (i.e., 9.5%) is similar to those made by using the k-nn (i.e., 10.2%) and the MLP (i.e., 10.4%) classifiers, it is the lowest one. This highlights that the proposed technique makes RBF classifiers a valid alternative to the nonparametric classifiers widely used in remote-sensing applications.

From the point of view of the processing time, the proposed training technique proved slightly faster than the classical one. In particular, in the experiments carried out, the training time was reduced by about 15%. This reduction in the training time was obtained by the simplification of the clustering problem associated with the presented technique.

In comparison with the other classifiers used in the experiments, the proposed technique required a much smaller training time than the MLP and SNN classifiers, whereas it proved slower than the PNN and k-nn classifiers.

VI. CONCLUSIONS

In this paper, we have presented a simple supervised technique for training the hidden layer of RBF neural network classifiers. The proposed technique, unlike traditional ones, selects the centers and widths of the kernel functions associated with the hidden neurons of the network by taking into account the class-membership information of training samples. In particular, such a technique avoids the generation of *mixed* kernel functions. In addition, it tunes the kernel function widths to limit the overlapping in boundary regions between different classes, while maintaining a certain overlapping inside each class. The proposed technique has significant advantages over traditional techniques:

1) increases the separability of classes in the kernel-function space and hence contributes to reducing the classification error made by the classifier;
2) reduces the oscillatory behavior of the classification error made by the network versus the number of hidden units,

thus providing a better framework for defining the network architecture;

3) improves the stability of the classification error of the network versus the random initialization of the kernel centers during the training process.

As a final remark, it is worth noting that, when the classes present a high degree of overlapping in the input space, the complexity of the classification problem may limit the capabilities of the proposed technique to reduce the overlapping of the kernel functions associated with the different classes. However, even in these extreme cases, the presented method prevents the generation of *mixed* clusters, whereas classical techniques would generate a large number of overlapping *mixed* clusters, which would strongly affect both the classification capability and the stability of the network.

Acknowledgment

The authors wish to thank Hunting Technical Services Ltd. for providing the remote-sensing images used in the experiments.

References

[1] J. Key, A. Maslanic, and A. J. Schweigner, “Classification of merged AVHRR and SMMR artic data with neural networks,” *Photogramm. Eng. Remote Sensing,* vol. 55, pp. 1331–1338, 1989.

[2] O. K. Ersoy and D. Hong, “Parallel, self-organizing, hierarchical neural networks,” *IEEE Trans. Neural Networks,* vol. 2, no. 2, pp. 167–178, 1990.

[3] J. A. Benediktsson, P. H. Swain, and O. K. Ersoy, “Neural network approaches versus statistical methods in classification of multisource remote sensing data,” *IEEE Trans. Geosci. Remote Sensing,* vol. 28, pp. 540–552, July 1990.

[4] J. D. Paola and R. A. Schowengerdt, “A review and analysis of backpropagation neural networks for classification of remotely-sensed multi-spectral imagery,” *Int. J. Remote Sensing,* vol. 16, no. 16, pp. 3033–3058, 1995.

[5] E. B. Baum, “On the capabilities of multilayer perceptrons,” *J. Complexity,* vol. 4, pp. 193–215, 1988.

[6] I. Kanellopoulos and G. G. Wilkinson, “Strategies and best practice for neural network image classification,” *Int. J. Remote Sensing,* vol. 18, no. 4, pp. 711–725, 1997.

[7] M. J. D. Powell, “Radial basis functions for multivariate interpolation: A review,” *Algorithms for Approximation,* J. C. Mason and M. G. Cox, Eds. Oxford, U.K.: Clarendon, 1987, pp. 143–167.

[8] D. S. Broomhead and D. Lowe, “Multivariate functional interpolation and adaptive networks,” *Complex Syst.,* vol. 2, pp. 321–355, 1988.

[9] J. Moody and C. J. Darken, “Fast learning in networks of locally-tuned processing units,” *Neural Computat.,* vol. 1, no. 2, pp. 281–294, 1989.

[10] C. M. Bishop, *Neural Networks for Pattern Recognition.* Oxford, U.K.: Clarendon, 1995.

[11] J. A. Benediktsson and J. R. Sveinsson, “Classification and feature extraction of AVIRIS data,” *IEEE Trans. Geosci. Remote Sensing,* vol. 33, pp. 1194–1205, Sept. 1995.

[12] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles.* Reading, MA: Addison-Wesley, 1974.

[13] S. B. Serpico, L. Bruzzone, and F. Roli, “An experimental comparison of neural and statistical nonparametric algorithms for supervised classification of remote-sensing images,” *Pattern Recognit. Lett.,* vol. 17, no. 13, pp. 1331–1341, 1996.