Proceedings of International Joint Conference on Neural Networks, Orlando, Florida, USA, August 12-17, 2007

A New Cost Function for Binary Classification Problems Based on the Distributions of the Soft Output for Each Class

Marcelino Lázaro, José M. Leiva-Murillo, Antonio Artés-Rodríguez, Aníbal R. Figueiras-Vidal

Abstract— **This paper proposes a new cost function for supervised training of neural networks in binary classification applications. This cost function aims at reducing the probability of classification error by reducing the overlap between distributions of the soft output for each class. The non parametric Parzen window method, with Gaussian kernels, is used to estimate the distributions from the training data set. The cost function has been implemented in a GRBF neural network and has been tested in a motion detection application from low resolution infrared images, showing some advantages with respect to the conventional mean squared error cost function and also with respect to the support vector machine, a reference binary classifier.**

I. Introduction

Artificial Neural Networks (NN) consist of two or more layers of non-linear units able to discover non-linear patterns present in the data. These networks have been successfully applied to both classification and regression problems. Classical NNs are built by means of units characterized by a sigmoid function. More recently, the radial basis function (RBF) has become popular as non-linear function for NNs. In this case, each unit is defined by a vector in which the *kernel* or spherical Gaussian function is centered. In the case of Generalized RBF networks (GRBF), the kernel is allowed to have a non spherical shape, and so a different width in each dimension. The RBF approach to learning is more related to the curve-fitting problem, so that the network is trained to interpolate the unknown classification/regression function [1].

Classical NNs may have one or more intermediate layers between the input and the output layers, like the network called multilayer perceptron (MLP). A multilayer network is more versatile than the single-layer network, so that it is more powerful at performing classification and regression tasks on data characterized by strongly non-linear patterns. Kolmogorov's Theorem says that any classification function can be learned from a training dataset for a MLP if the number of neurons in the hidden layer is big enough [2]. For RBF networks, another mathematical justification is provided by the Cover Theorem of separability of patterns [1]. However, the number of hidden units should be carefully chosen because a tradeoff between training error and generalization ability must be reached.

The Support Vector Machine (SVM) is considered as a new generation of Neural Networks. The most popular non-linear version of the SVM is the RBF-based one, so that its architecture is similar to the RBF network's. The functional to be minimized in SVM learning takes into account the complexity of the classification boundary, regularizing the discrimination function so that the solution is, in general, sparse. This way, a number of RBF units get a weight equal to zero, so that the number of neurons is set by the solver [3]. Another advantage with respect to NNs is the way SVMs penalize the error. Classical NN-based learning makes use of the mean square error (MSE) cost function to fit the weight and parameters of the neurons. Although this criterion makes sense in regression, it is not optimal in classification, because there is not a direct relationship between MSE and classification error. This problem is partially overcome by the SVM methodology, which considers an error based in the distance to the classification boundary if the sample is missclassified.

The effort to overcome the limitation of first and second order criteria in learning with NNs have given birth to the so-called information-theoretic learning (ITL) techniques. These methods are based on the application of Shannon's Information Theory concepts to the machine learning framework. In essence, ITL methods make use of cost functions based on the probability density functions (PDFs) at the output of a network. In binary classification, for example, it is desirable that the PDF of the two classes at the output are as less overlapped as possible. Renyi's entropy has been successfully used in unsupervised and supervised learning, together with PDF divergence measures based on the Euclidean distance or the Cauchy-Schwartz inequality [4]. These divergences provide alternative definitions of the mutual information (MI), which have been used in supervised linear feature extraction [5].

In this paper, we propose a method for learning with a GRBF by means of a cost function based on the distance between the PDF of each class at the output of the network. The PDFs are modeled by a non-parametric Parzen estimator, and the cost function has been chosen to be scale-invariant. A gradient descent based algorithm is proposed for the minimization of the cost and so the adjustment of the network parameters.

The paper is organized as follows. Section II presents the problem statement and introduces the notation employed in the paper. Section III presents the GRBF network architecture. Section IV introduces the proposed cost function, discusses its main properties and presents the implementation details of the iterative gradient algorithm used to minimize the cost. In Section V, we present the results obtained when

Authors are with the Departamento de Teoría de la Señal y Comunicaciones, Universidad Carlos III de Madrid, 29811, Leganés (Madrid), SPAIN (phone (first author): +34-91-624-8446; fax: +34-91-624-8749; email: {mlazaro, jose, antonio, arfv}@tsc.uc3m.es).

1-4244-1380-X/07/$25.00 ©2007 IEEE

the proposed cost function, implemented in a GRBF network, is used in a motion detection application from infrared images. Finally, Section VI comments the main conclusions about the proposed method.

II. Problem statement and notation

The problem we are facing in this paper is binary classification with neural networks in a supervised framework, which can be stated as follows: to train the network, a labeled data set of N samples is available

$$\{\mathbf{x}_k, z_k\},\ k = 1, \cdots, N,$$

with input patterns corresponding to a space input of dimension D, i.e., $\mathbf{x} \in I\!R^D$, and binary labels, $z_k \in \{0, 1\}$. We will denote by N_0 and N_1 the number of samples of class-0 ($z_k = 0$) and class-1 ($z_k = 1$), respectively. Of course, $N = N_0 + N_1$. Thus, the data set can be divided in two subsets, one for each class

$$\{\mathbf{x}_i^{(0)}\}, \{\mathbf{x}_j^{(1)}\},\ i = 1, \cdots, N_0, j = 1, \cdots, N_1.$$

A neural network will provide a soft output for each input pattern

$$y_k = h_{\mathbf{w}}(\mathbf{x}_k), \tag{1}$$

where vector $\mathbf{w}$ denotes the set of parameters of the neural network. Again, we can separate the soft outputs obtained from inputs of different classes using the notation

$$y_k^{(\ell)} = h_{\mathbf{w}}(\mathbf{x}_k^{(\ell)}),\ \ell \in \{0, 1\}.$$

Then, a hard output will be obtained from the soft one, selecting one of the possible classes for each pattern (not necessarily the correct one)

$$\hat{z}_i = g(y_k).$$

Network parameters $\mathbf{w}$ are selected to optimize a cost function measured over the available training data set with the goal of obtaining a good classification rate. We will discuss this point later in Section IV.

III. Generalized Radial Basis Function Network

In this paper, we concentrate on the application of the generalized radial basis function (GRBF) network, which is an extension of the radial basis function (RBF) network which allows a different variance for each input dimension [1]. The relaxation of the radial constraint transforms the standard Gaussian kernels with circular symmetry into elliptic basis kernels, which can reduce the number of necessary basis functions (or neurons) to adequately cover the input space for a given problem.

Specifically, we will work with a GRBF network with one hidden layer of N_n neurons and a linear output neuron, which provides the following output for a given D-dimensional input pattern

$$y_k = \sum_{n=1}^{N_n} \lambda_n o_n(\mathbf{x}_k), \tag{2}$$

where $o_n(\mathbf{x}_k)$ is the output of the n-th neuron in the hidden layer

$$o_n(\mathbf{x}_k) = \prod_{d=1}^{D} \exp\left\{-\frac{(x_{k,d} - \mu_{n,d})^2}{2\sigma_{n,d}^2}\right\},$$

and $\{\lambda_n\}$, are the weights of the linear output neuron. Therefore, the parameters of this network are the centers (D-dimensional) of each basis function, its variances or standard deviations (D-dimensional), and the N_n weights of the output neuron,

$$\mathbf{w} = [\boldsymbol{\mu}_1^T, \cdots, \boldsymbol{\mu}_{N_n}^T, \boldsymbol{\sigma}_1^T, \cdots, \boldsymbol{\sigma}_{N_n}^T, \lambda_1, \cdots, \lambda_{N_n}]^T, \tag{3}$$

where

$$\boldsymbol{\mu}_i^T = [\mu_{i,1}, \cdots, \mu_{i,D}]^T,\ \boldsymbol{\sigma}_i^T = [\sigma_{i,1}, \cdots, \sigma_{i,D}]^T.$$

IV. Cost function

In this section, first the proposed cost function will be introduced and discussed and then the implementation details will be presented.

A. Cost function and its properties

As previously mentioned in Section II, in a classification problem the neural network provides a soft output, which depends on the network parameters $\mathbf{w}$, as explicitly denoted in (1). In our case, the soft output is obtained by (2), and parameters are given in (3). Then, the classification label, 0 or 1, is obtained from the soft output.

In supervised training, parameters $\mathbf{w}$ are obtained by using the labeled training data set. A measurable (over the training data set) cost function is defined and some algorithm is applied to optimize such cost function. The most common approach consists in minimizing the mean squared error (MSE) between the soft output and a reference value (usually the known label) for each class. In this case, the cost function is

$$J(\mathbf{w}) = \frac{1}{N}\sum_{k=1}^{N}(y_k - z_k)^2.$$

This cost function can be minimized by a gradient descent algorithm, like in [6] for a GRBF.

To obtain the hard output, the soft one is compared with a threshold. Usually, the threshold is the mean value of the reference values for both classes, 0.5 if reference values are 0 and 1. Therefore

$$\hat{z}_k = g(y_k) = \begin{cases} 0, & \text{if } y_k > 0.5 \\ 1, & \text{if } y_k < 0.5 \end{cases}. \tag{4}$$

Although minimizing the MSE has shown a good performance over a great number of classification problems, this cost function was initially designed for regression problems. In classification problems, the main goal is not to minimize the MSE between labels and soft outputs but to minimize the probability of error. This probability is closely related to the distributions, for samples of each class, over the soft output y_k. If we denote by $f_{Y^\ell}(y)$ the distribution of the soft output

for input patterns of class ℓ, $\ell \in \{0,1\}$, the probability of classification error is

$$P_e = \pi_0 \int_{\mathcal{R}_1} f_{Y^0}(y)dy + \pi_1 \int_{\mathcal{R}_0} f_{Y^1}(y)dy,$$

where π_0 and π_1 are the probabilities of classes 0 and 1, respectively, and $\mathcal{R}_0$ and $\mathcal{R}_1$ are the decision regions for each class, i.e., $\mathcal{R}_\ell = \{y|g(y) = \ell\}$. To directly minimize this probability of error is not possible in most cases, but it is possible to design some cost functions aiming at reducing it in an indirect way.

Clearly, if distributions of both classes do not overlap and function $g(y)$ is properly designed, the probability of error becomes zero, while the probability of error grows when the overlapping between distributions increases. Therefore, cost functions measuring the overlapping between distributions for each class can be useful to reduce the probability of error.

In this paper we propose the following cost function to be minimized by the training algorithm of neural networks

$$J(\mathbf{w}) = \frac{\int_{-\infty}^{\infty} \hat{f}_{Y^0}(y) \cdot \hat{f}_{Y^1}(y)\, dy}{\int_{-\infty}^{\infty} \left(\hat{f}_{Y^0}(y) - \hat{f}_{Y^1}(y)\right)^2 dy},$$

where $\hat{f}_{Y^\ell}(y)$ denotes the estimate for the distribution of the soft output for class ℓ, which can be obtained from the available training data set. The inverse of the cost function can be regarded as a divergence measurement between estimated distributions, having the following properties:

- It is symmetric.
- Its maximum (minimum of the cost function) corresponds to distributions without overlap.
- It is scale invariant.

Another interesting characteristic of this cost function is that it is well defined for unbalanced training data sets, where the number of samples of each class is different. As long as the number of samples of each class is enough to perform a reasonable estimate for the distribution of its soft output, the relative number between samples of each class is not relevant.

It can be seen that both the numerator and denominator of the cost function could be a reasonable cost function by themselves. Looking at the numerator, it proposes to minimize the product of both distributions, which will tend to minimize the overlap between distributions. Looking at the denominator, it proposes to maximize the quadratic distance between distributions, which again will tend to "separate" distributions. The reason to avoid using any of these two terms independently is that both, numerator and denominator, are scale dependent, while the quotient is scale invariant. Therefore, using only one term can tend to scale the soft output to minimize (or maximize) the numerator (or denominator) instead of effectively reducing the overlap.

The non parametric Parzen window estimator [7] is used to obtain the distribution estimates by

$$\hat{f}_{Y^\ell}(y) = \frac{1}{N_\ell} \sum_{i=1}^{N_\ell} K_{\sigma_p}(y - y_i^{(\ell)}),$$

where $K_{\sigma_p}(y)$ has to be a valid probability distribution function and σ_p denotes its size. Here, we will use a Gaussian distribution with standard deviation σ_p.

The cost function can be rewritten as

$$J(\mathbf{w}) = \frac{\int_{-\infty}^{\infty} \hat{f}_{Y^0}(y) \cdot \hat{f}_{Y^1}(y)\, dy}{\int_{-\infty}^{\infty} \left[\left(\hat{f}_{Y^0}(y)\right)^2 + \left(\hat{f}_{Y^1}(y)\right)^2 - 2\hat{f}_{Y^0}(y)\hat{f}_{Y^1}(y)\right] dy}.$$

Taking into account that for Gaussian distributions

$$\int_{-\infty}^{\infty} K_{\sigma_p}(y-a) \cdot K_{\sigma_p}(y-b)\, dy = K_\sigma(a-b),$$

with $\sigma = \sqrt{2}\sigma_p$, the cost function becomes

$$J(\mathbf{w}) = \frac{A(\mathbf{w})}{B(\mathbf{w})} = \frac{A(\mathbf{w})}{B_0(\mathbf{w}) + B_1(\mathbf{w}) - 2A(\mathbf{w})},$$

where

$$A = \frac{1}{N_0 N_1} \sum_{i=1}^{N_0} \sum_{j=1}^{N_1} K_\sigma\left(y_i^{(0)} - y_j^{(1)}\right),$$

$$B_0 = \frac{1}{N_0^2} \sum_{i=1}^{N_0} \sum_{j=1}^{N_0} K_\sigma\left(y_i^{(0)} - y_j^{(0)}\right),$$

and

$$B_1 = \frac{1}{N_1^2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_1} K_\sigma\left(y_i^{(1)} - y_j^{(1)}\right).$$

B. Implementation details

The proposed cost function will be minimized by means of an iterative gradient descent algorithm. The gradient of the cost function can be written as

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \sum_{\ell=0}^{1} \sum_{j=1}^{N_\ell} \frac{\partial J(\mathbf{w})}{\partial y_j^{(\ell)}} \frac{\partial y_j^{(\ell)}}{\partial \mathbf{w}}.$$

In this case,

$$\frac{\partial J(\mathbf{w})}{\partial y_j^{(\ell)}} = \frac{\left(\frac{\partial A(\mathbf{w})}{\partial y_j^{(\ell)}} B(\mathbf{w}) - A(\mathbf{w}) \frac{\partial B(\mathbf{w})}{\partial y_j^{(\ell)}}\right)}{(B(\mathbf{w}))^2},$$

where

$$\frac{\partial A(\mathbf{w})}{\partial y_j^{(0)}} = -\frac{1}{N_0 N_1} \sum_{i=1}^{N_1} K_\sigma(y_j^{(0)} - y_i^{(1)}) \frac{y_j^{(0)} - y_i^{(1)}}{\sigma^2},$$

$$\frac{\partial A(\mathbf{w})}{\partial y_j^{(1)}} = \frac{1}{N_0 N_1} \sum_{i=1}^{N_0} K_\sigma(y_j^{(0)} - y_j^{(1)}) \frac{y_i^{(0)} - y_j^{(1)}}{\sigma^2},$$

$$\frac{\partial B(\mathbf{w})}{\partial y_j^{(0)}} = -\frac{2}{N_0^2}\sum_{i=1}^{N_0} K_\sigma(y_j^{(0)} - y_i^{(0)})\frac{y_j^{(0)} - y_i^{(0)}}{\sigma^2} + \frac{2}{N_0 N_1}\sum_{i=1}^{N_1} K_\sigma(y_j^{(0)} - y_i^{(1)})\frac{y_j^{(0)} - y_i^{(1)}}{\sigma^2},$$

and

$$\frac{\partial B(\mathbf{w})}{\partial y_j^{(1)}} = +\frac{2}{N_1^2}\sum_{i=1}^{N_1} K_\sigma(y_i^{(1)} - y_j^{(1)})\frac{y_i^{(1)} - y_j^{(1)}}{\sigma^2} - \frac{2}{N_0 N_1}\sum_{i=1}^{N_1} K_\sigma(y_i^{(0)} - y_j^{(1)})\frac{y_i^{(0)} - y_j^{(1)}}{\sigma^2}.$$

Finally, for the proposed GRBF network, whose soft output is given by (2), the partial derivatives of the soft output with respect to the network parameters are, independently of the class label

$$\frac{\partial y_k}{\partial \mu_{i,j}} = \frac{x_{k,j} - \mu_{i,j}}{\sigma_{i,j}^2}\lambda_i o_i(\mathbf{x}_k),$$

$$\frac{\partial y_k}{\partial \sigma_{i,j}} = \frac{(x_{k,j} - \mu_{i,j})^2}{\sigma_{i,j}^3}\lambda_i o_i(\mathbf{x}_k),$$

and

$$\frac{\partial y_k}{\partial \lambda_i} = o_i(\mathbf{x}_k).$$

Once the GRBF has been trained from data samples to obtain the $\mathbf{w}$ parameters, the classification has to be done in order to minimize the probability of error. Assuming that both classes have the same probability, the decision rule is

$$\hat{z_k} = g(y_k) = \begin{cases} 0, & \text{if } \hat{f}_{Y^0}(y_k) > \hat{f}_{Y^1}(y_k) \\ 1, & \text{if } \hat{f}_{Y^0}(y_k) < \hat{f}_{Y^1}(y_k) \end{cases}.$$

To implement this decision rule, it is not necessary to evaluate $\hat{f}_{Y^0}(y_k)$ and $\hat{f}_{Y^1}(y_k)$ each time a new sample is presented to the network. The simplest method is to evaluate both $\hat{f}_{Y^0}(y)$ and $\hat{f}_{Y^1}(y)$ as a function of y and establish the thresholds dividing the output space in decision regions for each class. This can be accomplished numerically in an efficient way from the training data set.

V. Application to People Motion Detection from Low Resolution IR Images for Security

We have tested the proposed cost function implemented in a GRBF network for a security application, to detect intrusion of people in a room that is monitored by an infrared (IR) digital camera. In this section, first we will describe the characteristics of the images to be classified, and then the simulation results obtained in the classification will be presented.

A. Characteristics of the IR images

The IR camera is a low cost and low resolution (16×16 pixels) prototype based on PbSe sensors. PbSe sensors have a relatively high responsitivity at room temperature, which allows to work without need for cooling, thus providing low cost detectors.

Based on the images provided by the IR camera, we desire to implement a binary classifier to decide if a person is or not present at each image. The classifier will trigger an event to record the video sequence when a target has been detected. However, the actual prototype suffers from a high thermal drift, which seriously difficults the design and development of a neural network based presence detector. To overcome the thermal drift, a differential scheme has been adopted. The camera does not provide to the classifier the acquired image but the difference between the actual image and the image obtained one second before. The thermal drift is slow enough to be neglected in this short interval. This means that the application becomes a motion detector because a static target will not be detected. However, since an intruder has to move to get into the room and to go out of the room, a motion detector will be enough to our requirements. Figure 1 shows the 16×16 IR image for a person. It can be seen the silhouette of the person and a shadow at the position where it was a second before. Figure 2 shows the differential IR image when no target is in scene. It can be seen that the low resolution image is enough to discriminate a person present in an image.

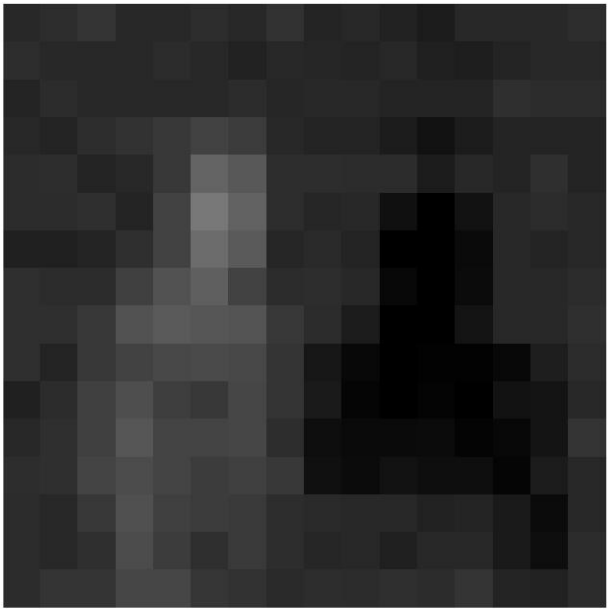

Fig. 1. 16×16 pixel differential IR image with a person.

B. Classification results

A labeled set of differential images has been acquired to train the binary classifier in a supervised training scheme. We have divided it in three subsets: training set (515 images), validation set (520 images), and test set (4632 images).

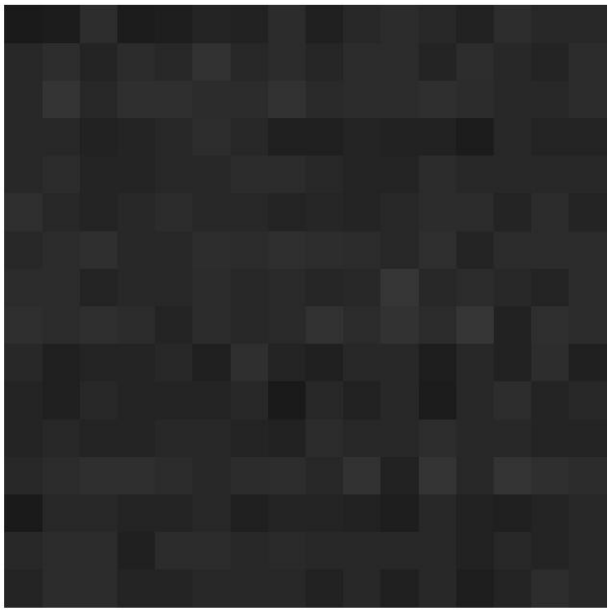

Fig. 2. 16×16 pixel differential IR image without any target.

To reduce the data dimensionality and to improve the generalization capability, an initial feature extraction is necessary. We have tested the performance in this application of the two most common blind techniques for feature extraction, Principal Component Analysis (PCA) [8] and Independent Component Analysis (ICA) [9]. PCA provided the best results and was finally selected. The number of components have been selected by cross-validation. Two components demonstrated to be enough to allow a suitable classification between the two specified classes in the problem at hand. After data normalization (between 0 and 1) PCA feature extraction with 2 eigenvectors obtained from the training data set was performed for the three subsets.

First, we have compared the performance of the proposed cost function, implemented in a GRBF network, with a SVM classifier. The LibSVM matlab package [10] has been used to obtain the SVM solution. Gaussian kernels have been selected. Once the kind of kernel has been selected, the SVM has two parameters: the weight C to ponder the classification errors, and the kernel variance. The optimum parameters have been obtained by cross-validation, giving $C = 0.9$ and $\sigma^2 = 0.05$ ($\gamma = 10$).

To provide a parsimonious solution, a GRBF with only 2 neurons was selected. To deal with the local minimum of the cost function, several independent trials with a random initialization were performed. More efficient initialization methods, as Orthogonal Least Squares (OLS) [11], could be employed. However, to evaluate the sensitivity of the method with respect to the local minima we preferred a random initialization. Centers of the Gaussian neurons ($\boldsymbol{\mu}_n$) are placed at the position of two input patterns randomly selected. Variances $\sigma_{n,k}$ were initialized with a uniform distribution in $[0.1, 1]$ independently for each neuron and dimension. Several sizes for the Parzen window kernel have been tested. Concretely

$$\sigma \in \{0.1, 0.25, 0.5, 1, 5, 10\}.$$

To prevent over-training, the validation set has been used to decide the end of the iterative updating algorithm.

Table I summarizes the results obtained with SVM and a GRBF with the proposed cost function. 300 experiments with different initializations were performed for the GRBF network with the proposed algorithm. The best SVM solution provides a higher probability of error in the classification of the test set with a higher number of support vectors (127). Therefore, the SVM method has 256 parameters (two coordinates per support vector, C, and the Gaussian kernel size). The proposed method, using only two neurons (10 parameters) is able to obtain a lower probability of error in this problem, while the conventional formulation of the SVM does not allow to fix the network size if performance is used (for instance by cross-validation) to obtain the solution.

TABLE I

COMPARATIVE OF RESULTS OBTAINED WITH A SVM AND A GRFB USING THE PROPOSED COST FUNCTION.

Network	Minimum Error	Neurons (SV's)	Parameters
SVM	0.90674 %	127	256
Proposed	0.7556 %	2	10

We also consider interesting the comparison with the same network architecture using a different training algorithm. We have compared the proposed cost function against the classical MSE cost function. The GRBF with MSE cost function was trained by the gradient descent algorithm presented in [6] and combined with a decision device given by (4). The same random initialization procedure detailed before was used for both algorithms and 300 trials were performed.

Figures 3 and 4 compare the error rate (in %) obtained with the MSE and the proposed cost function, respectively. For the proposed method, results obtained for the optimal size of the Parzen method ($\sigma = 5$) are plotted.

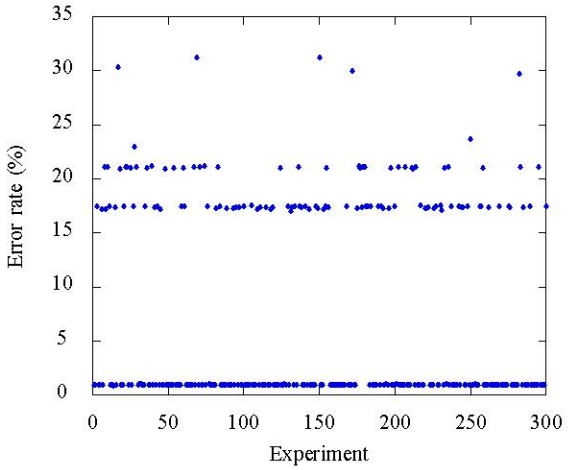

Fig. 3. Error rate for the MSE cost function in the 300 independent experiments.

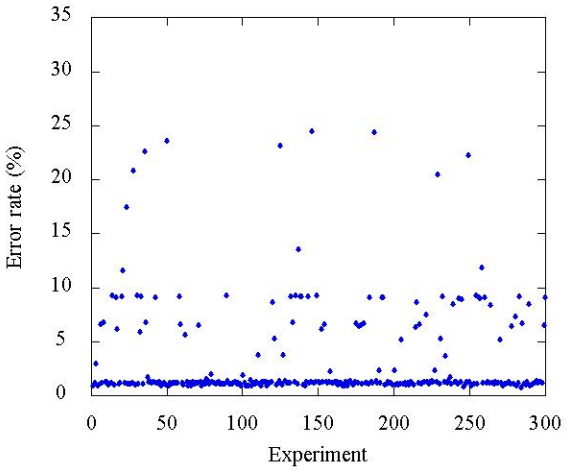

Fig. 4. Error rate for the proposed cost function in the 300 independent experiments.

The proposed method shows a lower sensitivity to local minima than the MSE cost function. MSE seems to be trapped at several local minima and seems to tend to some fixed solutions. Table II provides the minimum error rate, the mean value, and the standard deviation of the error rate. These measures are provided for the MSE cost function, for the proposed method using the optimal Parzen kernel size, and the proposed method averaging results with all kernel sizes. The proposed method provides a better error rate than the MSE cost function. The mean error rate obtained with the optimal kernel size is much lower than the obtained with the MSE cost function. More interesting is that even averaging results for all kernel sizes the proposed method has a lower mean value.

TABLE II

COMPARATIVE OF RESULTS OBTAINED WITH A GRFB USING THE MSE AND THE PROPOSED COST FUNCTIONS.

Cost function	Minimum Error	Mean Error	STD
MSE	0.8851 %	7.8573 %	9.0481
Proposed (Optimum σ)	0.7556 %	3.2464 %	4.3986
Proposed (All σ's)	0.7556 %	3.9773 %	4.4061

Considering the effect of the kernel size, simulations have shown that this parameter is not very critical. Table III compares the minimun, mean, and the standard deviation of the error rate for all values of the kernel size σ. Although the best results are obtained for $\sigma = 5$, the differences are small and with all values the best result is better than the obtained with the MSE cost function and the SVM.

TABLE III

RESULTS OBTAINED WITH THE PROPOSED COST FUNCTION FOR DIFFERENT PARZEN KERNEL SIZES.

σ	0.1	0.25	0.5	1	5	10
Minimum error	0.82	0.78	0.78	0.79	0.75	0.86
Mean error	5.46	4.08	3.81	3.88	3.24	3.36
STD	3.94	4.23	4.12	4.69	4.39	4.61

VI. CONCLUSIONS

A new cost function is proposed in this paper to train neural networks for problems of binary classification. The proposed cost function aims to reduce the overlap between the distributions of the soft neural network output for both classes, helping to reduce the probability of error. The Parzen window method is used to estimate the distributions from the training data set. Using Gaussian kernels for the Parzen methods brings out efficient adaptive equations to minimize the cost function by means of an iterative gradient descent algorithm. Furthermore, the cost function can be easily extended to multi-class classification problems.

The cost function is implemented in a GRBF network and it is applied to a motion detection security application from low resolution differential IR images. In this application, the proposed cost function and the GRBF architecture demonstrate to be able to obtain parsimonious solutions with better results than a more complex SVM solution, and than the same architecture trained with the conventional MSE cost function.

Further work is necessary to evaluate the performance of this cost function in other applications, with other network architectures, and to analyze alternative extensions and modifications of this proposed cost function, such as considering the distributions of each class in the outputs of the neurons in the hidden layer along with the distributions of the network output.

ACKNOWLEDGMENT

This work has been partly supported by Ministerio de Educación y Ciencia of Spain (project 'DOIRAS', id. TIC2003-02602, and project 'MONIN', id. TEC2006-13514-C02-01), and Comunidad de Madrid (project 'PRO-MULTIDIS-CM', id. S0505/TIC/0223).

REFERENCES

[1] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 2nd edition, 1998.

[2] Christopher M. Bishop, *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995.

[3] B. Scholkopf and A. J. Smola, *Learning with Kernels*, The MIT Press, Cambridge, MA, 2002.

[4] J. Principe, D. Xu, and J. W. Fischer III, *Information-Theoretic Learning*, vol. 1, Wiley, 2000.

[5] K. Torkkola, "Feature extraction by non-parametric mutual information maximization," *Journal on Machine Learning Research*, vol. 3, pp. 1415–1438, 2003.

[6] I. Santamaría, M. Lázaro, C. J. Pantaleón, J. A. García, A. Tazón, and A. Mediavilla, "A nonlinear MESFET model for intermodulation analysis using a generalized radial basis function network," *Neurocomputing*, vol. 25, pp. 1–18, 1999.

[7] E. Parzen, "On the estimation of a probability density function and the mode," *Annals of Mathematical Statistics*, vol. 33, pp. 1065–76, 1962.

[8] Ian T. Jollife, *Principal Component Analysis*, Springer-Verlag, New York, 2nd edition, 2002.

[9] A. Hyvarinen and E. Oja, "Independent component analysis: Algorithms and applications," *Neural Networks*, vol. 13, pp. 411–430, 2000.

[10] C. Chang and C. Lin, "Libsvm: a library for support vector machines," software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm/, 2001.

[11] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis functions," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 302–309, Mar. 1991.